

Bots In Brief - What's New • What's Cool • What's Weird

SERVO

FOR THE ROBOT INNOVATOR

www.servomagazine.com

MAGAZINE

February 2011

Build A PS2 Servo CONTROLLER Interface

♦ **Take A Look
At The Robotis
OLLO Explorer**
A great starter kit
for kids of all ages.

♦ **Building A Human
Sized Walker**
Taking the first steps
to creating a full size
humanoid robot.

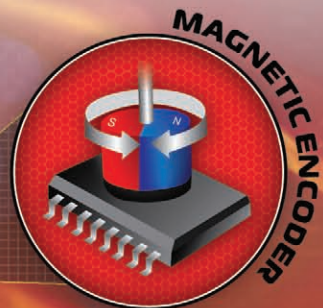
♦ **Ask Mr. Roboto**
♦ **Robots Then & Now**



INVEST in the BEST

YOUR ROBOT WILL THANK YOU

At Hitec, we know how much time and money you dedicate to your robot hobby. So why not make sure your servos are delivering what they promise? Our technologically advanced mega servos are tough enough for even your most challenging projects. Designed with our newest high resolution "G2.5" 12-bit programmable digital circuit and indestructible titanium gears, the HS-7980TH and HS-M7990TH give mega torque and speed with pinpoint accuracy. Built to last, these servos bring unprecedented power and sustainability to your investment.



Our HS-7980TH employs durable mechanical potentiometer technology while the HS-M7990TH utilizes the first ever magnetic encoder.



Model	6 Volts		7.4 Volts		Part#	Dimensions	Weight
	Speed	Torque	Speed	Torque			
HS-7980TH	0.20	500 oz.in	0.17	611 oz.in	37980S	1.72 x 0.88 x 1.57 in	2.70 oz
HS-M7990TH	0.20	500 oz.in	0.17	611 oz.in	37990S	1.72 x 0.88 x 1.57 in	2.70 oz



We Serve you Right!

12115 Paine Street, Poway, CA 92064 • 858-748-6948 • www.hitecrd.com

Thanks to Lynxmotion for the Phoenix robot featured in this advertisement.

METAL GEARMOTORS

ORANGUTAN
CONTROLLER BOARD

60MM WHEELS

CHECK AT POLOLU.COM!

NEEDED:

LASER-CUT CUSTOM PIECES?
INERTIAL SENSORS?
MOTOR CONTROLLERS

★ REMEMBER TO USE
COUPON CODE SERVO339

SERVO

LEDS

LCD DISPLAY

AAA BATTERIE

Take your design from
idea to reality

SERVO

MAGAZINE

02.2011

VOL. 9 NO. 2

Did you know that each article in *SERVO Magazine* has its own webpage? It's where you go for downloads, comments, updates, corrections, or to link to the article in the digital issue. The unique link for each webpage is included with the article.

You can also visit article pages from back issues at www.servomagazine.com. Just select the **Past Issues** tab from the **About SERVO** drop down menu, click the Table of Contents link, and the article name.

Columns

- 08 Robytes**
by Jeff Eckert
Stimulating Robot Tidbits
- 10 GeerHead**
by David Geer
Simple Spine of Lamprey Leads to Complex Robot Limbs and Neuroprostheses
- 13 Ask Mr. Roboto**
by Dennis Clark
Your Problems Solved Here
- 76 Then and Now**
by Tom Carroll
Amazing Robots Arise From Junk



Departments

- | | |
|---------------------------|------------------------------|
| 06 Mind/Iron | 20 Bots in Brief |
| 15 Showcase | 64 SERVO Webstore |
| 16 Events Calendar | 81 Robo-Links |
| 18 New Products | 81 Advertiser's Index |

The Combat Zone...

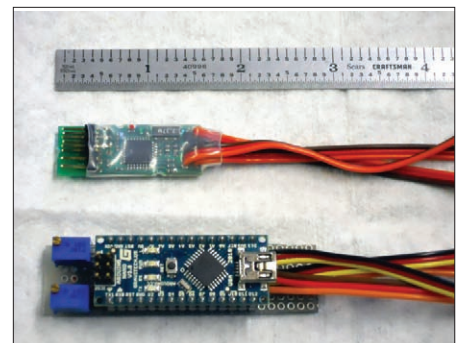
Features

- 26 PARTS IS PARTS:**
DeWalt Drill Motor Mounting Solution
- 27 MANUFACTURING:**
Closed Loop Lifter Solutions
- 30 Potpourri**

Events

- 32 EVENT REPORT:**
Full Metal Carnage 2 Pushes Robot Combat Over the Edge
- 33 Upcoming Events**

PAGE 27



SERVO Magazine (ISSN 1546-0592/CDN Pub Agree#40702530) is published monthly for \$24.95 per year by T & L Publications, Inc., 430 Princland Court, Corona, CA 92879. PERIODICALS POSTAGE PAID AT CORONA, CA AND AT ADDITIONAL ENTRY MAILING OFFICES. POSTMASTER: Send address changes to **SERVO Magazine, P.O. Box 15277, North Hollywood, CA 91615** or Station A, P.O. Box 54, Windsor ON N9A 6J5; cpcreturns@servomagazine.com

In This Issue ...

34 The Robotis OLLO Explorer

by *Tom Carroll*

Take an indepth look at this great robot starter kit for kids of all ages.

37 Intelligent Vacuum Cleaner Systems

by *John Blankenship and Samuel Mishal*

Explore advanced behaviors in robotic commercial products through simulation.

40 The NXT Big Thing #7

by *Greg Intermaggio*

Eddie 2.0 does Sumo.

47 Build Your Own Big Walker

by *Daniel Albert*

If you've ever wanted to build a full size humanoid robot, here's your chance. Part 1 will get you started with some basic ideas and concepts.

50 Control Your Steel Army With the RPM3

by *Fred Eady*

The Radio Packet Modem 3 is not your average embedded radio. You can communicate with one steel soldier or a whole platoon.

57 Playstation Servomotor Controller Interface

by *John Iovine*

This interface allows you to control six hobby servomotors using a PS2 controller.

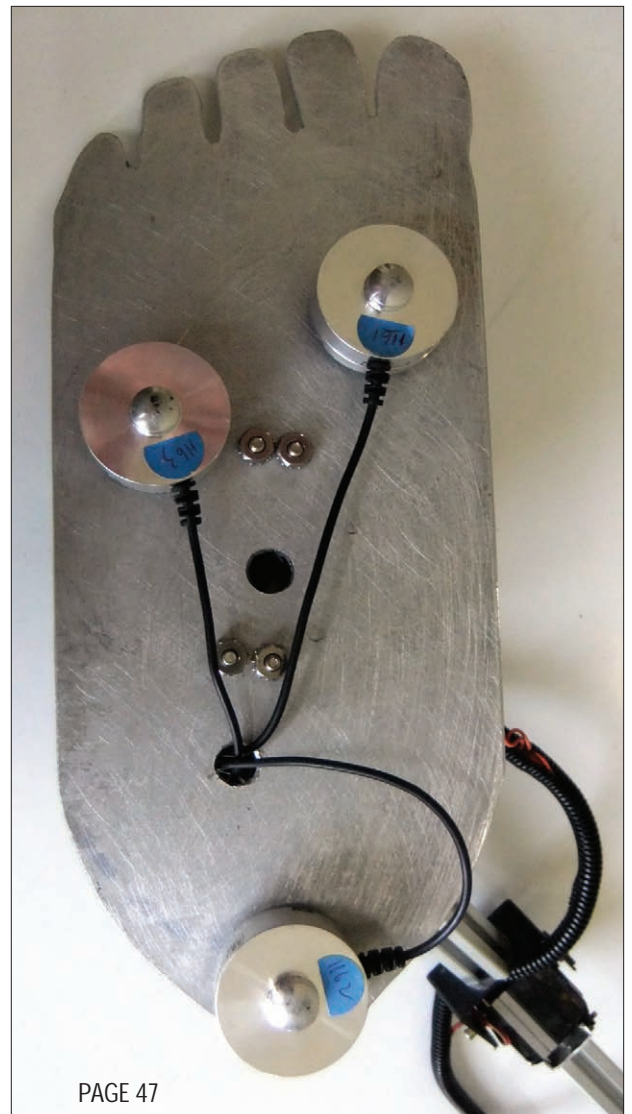
67 Making Robots With the Arduino — Part 4

by *Gordon McComb*

Getting Feedback With Sensors.



PAGE 34



PAGE 47

Mind / Iron

by Bryan Bergeron, Editor

When To Move On

The quest for innovative solutions in robotics sometimes requires taking the painful step of leaving behind familiar technology in favor of new, unproven technology. The hope is that the return on investment of moving forward pays for itself in some combination of new functionality, smaller form factor, less cost, lighter weight, or ease of development. Sometimes this bears out, and sometimes it doesn't.

For example, I recently moved from a line of microprocessor that I've used for years in favor of a new processor with a more powerful architecture. The original microprocessor line hasn't been upgraded in years, and relative performance just isn't in line with what's available on the market today. So far, the cost of moving has been significant, but bearable.

The main issue in working with a new microprocessor, of course, is software. I had to learn several new software tools and the nuances of debugging in a new architecture. In my case, the Achilles heel of the move is the lack of a powerful debugging environment. Instead of breakpoints and register displays, I'm working with LEDs and an interactive terminal. It's workable, but not as efficient as my old debugger.

Then, there's the constant temptation to revert to my old platform. For example, my latest project involved interfacing a GPS module with a navigation system. Even though I had the code in a library for my old processor, I spent three days with the new processor. I wrote off the time as a learning experience, but it was tempting to use the old tools just one more time. Of course, I realized that doing so would just prolong my learning time with the new processor. The move paid off. The navigation system has more processing power and better network connectivity than I could have had with my old hardware.

While processor moves are a major commitment in time and resources, sensors, actuators, and other

components tend to be no-brainers. As long as the device sports a standard interface, it's often simply a question of relative cost and performance. For example, I recently made the move from mechanical rotary switches to rotary encoders. I had resisted the move for years, simply because of the software and processing overhead of dealing with an encoder. However, when a recent project required a 12-position rotary switch, it became a question of whether to use a mechanical rotary switch at a dollar per contact versus a rotary encoder at 25 cents per contact. The choice of a rotary encoder was easy, especially given that a dozen units were involved.

Other device moves may be less straightforward. For example, the same project with the rotary switch involved an ultrasonic rangefinder. I've been using dual-element rangefinders sold by Acroname (www.acroname.com) and Parallax (www.parallax.com) for years, but there simply wasn't room in the new project for separate transmit and receive elements. Instead, I opted for a single element transceiver to save space and a few grams. The cost involved learning a new detection map. Instead of working intuitively with the sensor layout, I had to refer to the map provided by the rangefinder manufacturer to make certain I didn't have any blind spots in rangefinder coverage.

The question of whether to stay put or move on to a new technology platform is becoming more frequent, thanks to the increasing number of options in the market. In particular, I've been watching the evolution of the Intel Atom Processor (www.intel.com/technology/atom), especially the low power version designed for smartphones and tablets. With up to 1M of cache and 2.13 GHz operation, it certainly blows away my current processor. However, moving to the Atom platform would entail another round of learning the hardware, software tools, and limitations. For now, I'm staying put, simply to avoid the churn. **SV**

FOR THE
ROBOT
INNOVATOR

SERVO
MAGAZINE

Published Monthly By
T & L Publications, Inc.
430 Princeland Ct., Corona, CA 92879-1300
(951) 371-8497
FAX (951) 371-3052
Webstore Only **1-800-783-4624**
www.servomagazine.com

Subscriptions
Toll Free **1-877-525-2539**
Outside US **1-818-487-4545**
P.O. Box 15277, N. Hollywood, CA 91615

PUBLISHER
Larry Lemieux
publisher@servomagazine.com

**ASSOCIATE PUBLISHER/
VP OF SALES/MARKETING**
Robin Lemieux
display@servomagazine.com

EDITOR
Bryan Bergeron
techedit-servo@yahoo.com

CONTRIBUTING EDITORS
Jeff Eckert
Tom Carroll
Dennis Clark
Fred Eady
Gregory Intermaggio
Gordon McComb
John Iovine
Don Hebert
Jenn Eckert
David Geer
R. Steven Rainwater
Kevin Berry
John Blankenship
Samuel Mishal
Daniel Albert

CIRCULATION DIRECTOR
Tracy Kerley
subscribe@servomagazine.com

**MARKETING COORDINATOR
WEBSTORE**
Brian Kirkpatrick
sales@servomagazine.com

WEB CONTENT
Michael Kaudze
website@servomagazine.com

ADMINISTRATIVE ASSISTANT
Debbie Stauffacher

PRODUCTION/GRAPHICS
Shannon Christensen

Copyright 2011 by
T & L Publications, Inc.
All Rights Reserved

All advertising is subject to publisher's approval. We are not responsible for mistakes, misprints, or typographical errors. *SERVO Magazine* assumes no responsibility for the availability or condition of advertised items or for the honesty of the advertiser. The publisher makes no claims for the legality of any item advertised in *SERVO*. This is the sole responsibility of the advertiser. Advertisers and their agencies agree to indemnify and protect the publisher from any and all claims, action, or expense arising from advertising placed in *SERVO*. Please send all editorial correspondence, UPS, overnight mail, and artwork to: **430 Princeland Court, Corona, CA 92879.**

Printed in the USA on SFI & FSC stock.



X-TREME geek

x-treme Geek is wired
in to what you want.

You want that Big Reaction when your loved one opens their gift. We guarantee you'll get it when you shop with us. From never before seen modern marvels to kitschy cool blasts from the past, X-treme Geek has the gifts to make this holiday the best one yet. Shop x-tremegeek.com.

2500941.....\$16.95

EchoBot Voice Messenger

- Detects movement up to 3 feet away



Computer not included

EchoBot Delivers a Voice Message when Somebody Moves Nearby

Record messages for your friends and coworkers, and the EchoBot will play them back when they approach. Up to ten seconds of audio will play back when someone triggers the motion detector (built into the "eye," of course). Bendable legs and suction cup feet make it easy to position the EchoBot anywhere around your home, office or car. Colors vary. Size: 7" tall.

A Carpentry Kit for a Robot?

NEW

Maybe early robots were made of wood....or maybe it's just an awesome idea who's time is long overdue. This kit is supplied with pre-punched wooden boards, gears, shafts, switch, motor, battery holder, and all necessary parts to build your own wooden robot, complete with blinking eyes. Includes easy-to-follow instructions. Requires screwdriver and long nose pliers for construction, plus two AA batteries - all not included. Recommended for ages 10 and up.



3152147.....\$19.95

Robomech Wooden Kit

WARNING:
CHOKING HAZARD—Small Parts.
Not for children under 3 yrs.

POW Robot T-Shirt

NEW

From R2-D2 and C3-PO to Johnny 5 and Rosie, there are few things cooler than robots. Now you can proudly sport your robot fandom in this cotton tee. Accented by comic book style POW background, the robot pictured on this shirt will WOW your friends while you look cool, calm, and comfortable. Made in America and printed with environmentally friendly inks. Color: Heather



POW Robot \$24.95

- 3200150.....Medium
- 3200151.....Large
- 3200152.....X-Large
- 3200153.....XX-Large

Virtual Guard Dog Protects Your Home 24/7



Dog not included

Switch from barking to tranquil rainforest sounds for a soothing audio backdrop for a party!

1320672.....\$89.95

Rex Plus, The Electronic Watchdog

- Adjustable volume & sensitivity
- 5 1/4" L x 5 3/4" W x 7 1/2" H; 6' cord

Few security systems are as intimidating as an angry barking dog. The Rex Plus security system "sees" through walls and doors to detect when someone approaches your home and then it "barks" just like a real German Shepherd. The startlingly realistic barking increases in intensity as the interloper gets closer. You get inexpensive, reliable, 24-hour protection that's perfect for extended absences.



Expecto Patronum!

NEW

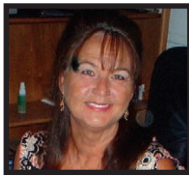
3200103.....\$89.95

The Wizard's Wand Universal Remote

Cast A Magic Spell Over Your Electronics!

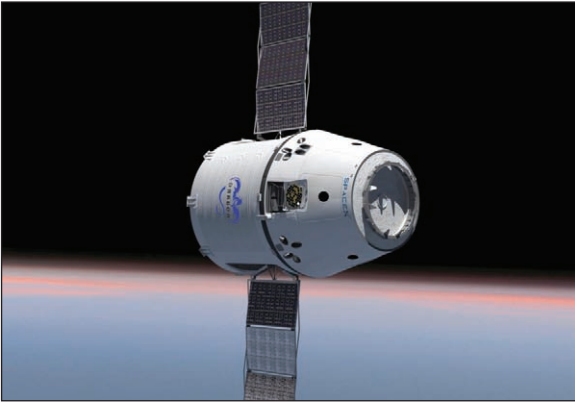
To truly rule the room and couch kingdom you must have a wizard on your side...or better yet, be a wizard yourself. With this vibrating wand as universal remote, controlling your home entertainment systems is simple sorcery. A flick of the wrist and a spin of your magical wand changes the channel, volume, track, and more, including rewind and fast forward. A total of 13 programmable commands are controlled by circular movements, up and down gestures, or back and forth whisks of this vibrating wand. It looks like magic, but it's really technology — the same accelerometer technology used in Wii remotes. With practice you'll master a level of wizard like skill to impress all your friends. The remote is recognized by almost every piece of modern home entertainment apparatus. Size: 39cm x 6.5cm x 3.8cm. Weight: 295g.

800.480.4335 • x-tremegeek.com



Robytes

by Jeff and Jenn Eckert



Dragon spacecraft with solar panels deployed.

Autonomous Cargo Spacecraft Launched

With the final Space Shuttle mission looming, its likely replacement — the SpaceX Dragon — lifted off from Cape Canaveral on December 8th atop a Falcon 9 launch vehicle. This was the first of at least 12 planned missions related to resupplying the International Space Station, although the December outing was just a short test flight aimed at demonstrating the system's ability to launch, orbit, communicate and maneuver, and return safely. An actual cargo mission isn't scheduled to happen until the third flight, later this year. Perhaps most notable is that Dragon can operate without human intervention, being capable of fully autonomous rendezvous and docking. However, it can be reconfigured to support up to seven passengers in the crew configuration, with members taking over control from the flight computer if desired. Also worth mentioning is that its \$1.6 billion budget covers a minimum of 12 flights, whereas a

single Shuttle launch runs about \$450 million. In case your local TV station didn't provide details, we can reveal that the vehicle can deliver payloads of up to 6,000 kg (13,228 lb) and bring back half that amount. In terms of volume, you can stash up to 10 cubic meters (245 cubic feet) in pressurized mode, and 14 cubic meters (490 cubic feet) unpressurized, inside. Lest you think this isn't a true commercial enterprise, note that non-ISS flights are available under the name DragonLab, so if you want to hire the platform for technology demonstrations, instrument tests, or whatever, all you need is a fat wallet. For details, visit www.spacex.com.

Flying Bot Dodges Obstacles

Another flying robotic platform — but much closer to home — is the Pelican, developed at the University of Pennsylvania (www.upenn.edu) and demonstrated at the 27th annual Army Science Conference in Orlando, FL. Pelican was created as part of the Army's Micro Autonomous Systems Technologies Collaborative Technology Alliance in an effort to create vehicles that can "think" their way through complex environments. It reportedly wowed the attendees by perfectly navigating through an obstacle course, going over, under, and around obstacles in its way.

Equipped with various sensors, a laser camera, and a low power computer to interpret what it "sees" and "feels," it can operate both indoors and out. In spite of its



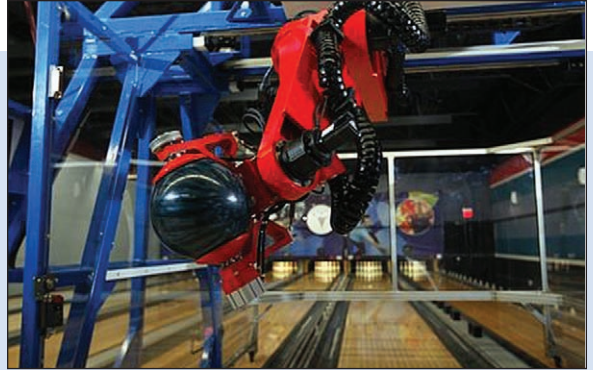
The Pelican flying robot navigates over, under, and through obstacles. Photo by Sarah Maxwell, Army Research Lab.

operational complexity, the bot weighs less than three lb. According to Penn's Dr. Nathan Michael, the computer constantly asks such questions as "Where is my location? How do I plan in that map? How much wind can I control?" and then determines its own route by taking into account the environment, what it needs to do to keep flying, and even its battery level.

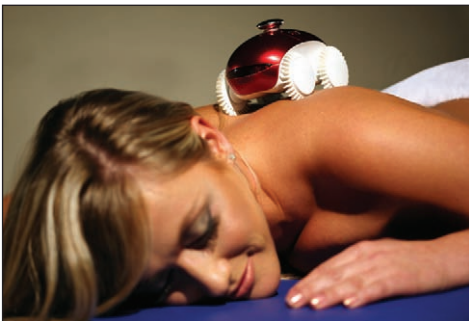
Future versions may be scaled back to fly bots small enough to fit in the palm of your hand. The long-term goal of the Alliance is "to develop autonomous, multifunctional, collaborative ensembles of agile, mobile micro-systems to enhance tactical situational awareness in urban and complex terrain for small unit operations."

My Name Is E.A.R.L.

No, this has nothing to do with karma, unless you have a spiritual bond with bowling. E.A.R.L. stands for Enhanced Automated Robotic Launcher, a name somewhat tortuously devised to honor bowler Earl Anthony, whose nickname was "The Machine." It's a state-of-the-art bowling bot designed to "consistently simulate any type of bowling style with an accuracy and consistency on the lanes that no human bowler can achieve. Those qualities make E.A.R.L. invaluable in the many studies necessary to keep up with the ever-changing bowling ball industry." Indeed, E.A.R.L. can throw right- or left-handed shots at speeds from 10 to 24 mph and with spin rates from 50 to 900 rpm, and do so consistently, shot after shot. It also sports a computer and sensor system that tracks the ball's location and speed as it moves down the lane, providing sophisticated ball-motion data. Although E.A.R.L. has been known to bowl a 300, he was pitted against pro bowler Chris Barnes last year, and Barnes beat the bot 259 to 209. According to the US Bowling Congress (www.bowl.com) — which uses E.A.R.L. for equipment specifications and certifications — the relatively poor performance comes because the bot "lays the ball down in exactly the same spot every time, so after only a couple of shots, the oil changes, and it's hard to strike consistently." Yeah, excuses, excuses.



The USBC's E.A.R.L. bowling bot.



A WheeMe massage bot gives "Sharon" the treatment.

WheeMe the Massage Bot

This month's most dubious robotic application takes the form of WheeMe — a palm-size bot that "gently massages and caresses as it moves slowly across your body." Slowly is right, with a top speed of only 1.6 ips. Given that it only weighs about 11.5 oz (330 g), you probably shouldn't expect it to feel much like a deep tissue massage applied by Otto the Animal or even a Swedish massage from Richard Simmons. Maybe the nylon fingerettes can penetrate deep enough to have some effect. At least the gadget employs sensors that allow it to steer itself over your body without falling off or losing its grip. If you want to try one out, just log onto www.dreambots.com and be willing to pay \$49.

Pachyderm-Inspired Handler

The Bionic Handling Assistant (BHA) — developed by Festo (www.festo.com) and introduced at the 2010 Hanover Fair — should look familiar, especially if you have been to the circus recently. Billed as an entirely new type of biomechatronic handling system, its design was inspired by the workings of an elephant's trunk. Notably, it differs from the usual heavy industrial robot in that machine-human contact is no longer hazardous to the latter; in the event of a collision, the system immediately yields without becoming unstable. In fact, the device is so revolutionary that its designers received the "Deutscher Zukunftspreis 2010" ("German Future Prize"), worth EUR 250,000 last December. According to a press release, "there are roughly 40,000 muscles that make the elephant's trunk an extremely flexible gripping hand that can move freely in any direction and can even rotate. A similar level of flexibility and gentle performance is provided by the high-tech trunk developed by researchers at Festo ..." The device consists of three basic elements: the hand axis, a ball joint, and the adaptive FinGripper which results in "smooth movement, more degrees of freedom (11), and an unparalleled mass/payload ratio." Weighing only 1.8 kg (4 lb), the plastic arm can lift up to 500 g (17.6 lb). Being powered by compressed air instead of mechanical devices, it is also relatively simple. Interestingly, the arm is created by a 3D printing process, so there is no assembly — all components are formed in a single step. The BHA doesn't appear in the current Festo catalog, but presumably it will be commercially available in the near future. **SV**



Festo's Bionic Handling Assistant, inspired by an elephant's trunk.



GEER HEAD

by David Geer

Contact the author at geercom@windstream.net

Simple Spine of Lamprey Leads to Complex Robot Limbs and Neuroprostheses

Researchers from Tulane and the University of Maryland along with Dr. Avis Cohen are working with a computational model of the Lamprey fish. The model consists of a system that solves several sets of equations around fluid motion and how the fish's muscles operate. "This is running on a cluster system at Tulane. It runs simulations," states Dr. Eric Tytell of Johns Hopkins.

Using this model, the researchers intend to study the Lamprey's simple spinal cord, in order to better understand the human spinal cord – which is not so simple.

Muscles are capable of a certain amount of force depending on how long the muscle is and how fast it is trying to contract. This creates mechanical feedback before the connection to the nervous system is made, according to Dr. Tytell.

A study of this force supplied by the Lamprey's muscles as it works with and against the force of the water

(fluid dynamics) is a large part of the research which should contribute to improved neuroprostheses and robotic limbs.

The Lamprey

The structure of the Lamprey nervous system, including the spinal cord, is basically the same as our human spinal cord, explains Dr. Eric Tytell, assistant research scientist on mechanical engineering at Johns Hopkins. "But, it is much simpler, so we can look at specific cells in detail to understand the changes that occur in an injured spinal cord in the Lamprey," comments Dr. Tytell.

"With the Lamprey, we know the cells that make up the neural circuit that controls the swimming and we know the first connections those cells make," Dr. Tytell continues.

With the information gained from this research, the scientists can focus on mechanical feedback instead of the sensory feedback which may not be available in a human being with spinal cord damage who needs a quality neuroprosthetic. By tailoring mechanical feedback, a prosthetic could do a lot of the things a normal limb can do even without all the sensory feedback.

As an example, a car's mechanics respond well to potholes even though there is no brain or sensory feedback available to the mechanics, Dr. Tytell explains.



Images of the Lamprey – the type of fish selected to create the computational model and framework for furthering research and development of neuroprostheses and robotic limbs.

Another angle on the Lamprey fish. Researchers Cohen and Tytell selected it for this research due to the simplicity and completeness of the current understanding of its spinal cord. Pictured are two silver Lampreys (*Ichthyomyzon unicuspis*), the most basal extant vertebrates which were used as the basis for a new computational model of swimming animals. The model accounts for both internal muscle mechanics and external fluid dynamics. It indicates that the tuning of biomechanical parameters such as body stiffness is critical for effective swimming performance in fishes.

"In the same way, the human body responds mechanically to the environment with no intervention of the nervous system. With prosthetics, you lose some of that nervous system connection. So, if we can determine how to properly tune those mechanics, enabling the fish to respond or people to respond mechanically, that may help design prosthetics that respond better even without input from the nervous system," says Dr. Tytell.

The scientists also intend to create controllers that mimic some of the assumed input of the nervous system to help neuroprostheses mimic the physical behavior that would be expected. "If we understand the circuitry in a spinal cord that moves your legs, maybe we can build controllers based on that circuitry. We can stimulate the muscles in a paraplegic to move electrically but we still need to control the process somehow. We can mimic how the spinal cord does it by looking at the Lamprey which has a spinal cord that is easier to study," says Dr. Tytell.

The study of how the fish muscles interact with the water provides some of this mechanics intelligence. The researchers have a theory and about 1-1/2 equations that analyze the mechanical interactions between the muscles and the fluid environment. This provides a framework for interpreting how animals and people interact with their physical environments, including how a person with a prosthetic limb might interact with the surface they walk on.

As a car's suspension system provides mechanical stability to help the car negotiate the potholes, the Lamprey has mechanical stability that the scientists need to investigate to transfer that capability to people walking with prosthetics.

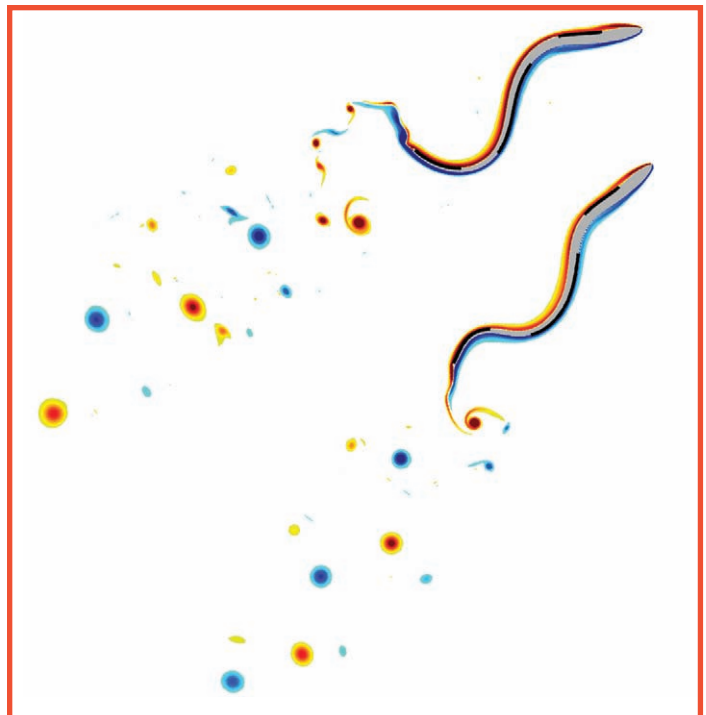
Lessons So Far

At this point, the researchers have learned that body



stiffness alone is not a determinate of how fast varying fish can accelerate. Acceleration seems to be a combination of body stiffness and muscle strength, Dr. Tytell explains.

"If the body is stiff and the muscles are strong, then you end up with high acceleration. With the computational model of the swimming fish, we did something that evolution would never do: We made a



This model shows the water flow in the wakes behind swimming Lampreys. The model accounts for both internal muscle mechanics and external fluid dynamics. Red and blue colors indicate the amount the fluid is rotating (called vorticity); active muscles are shown with thick black lines. The upper swimmer has a stiff body and strong muscles which allow it to accelerate faster, while the lower one has a floppy body and weak muscles, and is more energy efficient.

GEERHEAD

simulation of a stiff-bodied animal with weak muscles. This way, we learned that if everything is constant in a weak body it will bend more, and that larger bend causes greater acceleration," Dr. Tytell continues.

The researchers are also testing the hypothesis that matching the mechanical properties of future prosthetic devices to the body's natural mechanics will help produce better, more natural prosthetics. "We are aiming at the mechanical stability of the body. We want to help develop appropriate mechanics for prosthetics and have those be similar to body mechanics so they might be able to do similar things," Dr. Tytell explains.

The problem with older prosthetics is that they work against the body's natural mechanics rather than with them. "Newer prosthetics take this more into account. If you design something heavy and stiff like a robot leg, that won't work. The Asimo robot, for example, is a very stiff, strong, and heavy robot. In our legs, our bones are stiff but the joints are compliant. Older prosthetics try to control everything rather than letting some of the movement be dictated by the natural movement of the system," says Dr. Tytell.

Future Applications

"If we understand how animals control movements in unpredictable environments, we can design prosthetics with control systems, robot limbs, or an autonomous submarine, for example, that can do some of the same things a fish can do," concludes Dr. Tytell. **SV**

Resources

Dr. Eric Tytell


<https://limbs.lcsr.jhu.edu/User:ETytell>

Dr. Avis Cohen

www.isr.umd.edu/faculty/gateways/acohen.htm

FUN_{DAMENTALS} For The Beginner

Need the Basics?
Follow along with a new series of articles,
(which including easy to understand graphics)
Starting in the May 2010 issue
of
Nuts & Volts Magazine

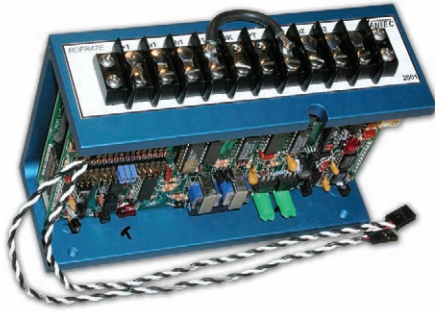


Schematic

Note: Electron flow is from the cathode of the LED to the anode. If you connect the LED to the circuit and LED VOLT, NOCT ADAPT UP

Subscribe Today! (with optional digital back issue viewing.) visit www.nutsvolts.com or call (800) 783-4624

STEER WINNING ROBOTS WITHOUT SERVOS!



Perform proportional speed, direction, and steering with only two Radio/Control channels for vehicles using two separate brush-type electric motors mounted right and left with our **mixing RDRF dual speed control**. Used in many successful competitive robots. Single joystick operation: up goes straight ahead, down is reverse. Pure right or left twirls vehicle as motors turn opposite directions. In between stick positions completely proportional. Plugs in like a servo to your Futaba, JR, Hitec, or similar radio. Compatible with gyro steering stabilization. Various volt and amp sizes available. The RDRF47E 55V 75A per motor unit pictured above.
www.vantec.com

VANTEC

Order at
(888) 929-5055

THE ORIGINAL SINCE 1994

PCB-POOL[®]

Beta LAYOUT

Servicing your complete PCB prototype needs :

- **Low Cost - High Quality**
PCB Prototypes
- **Easy online Ordering**
- **Full DRC included**
- **Lead-times from 24 hrs** NEW!
- **Optional Chemical Tin finish**
no extra cost

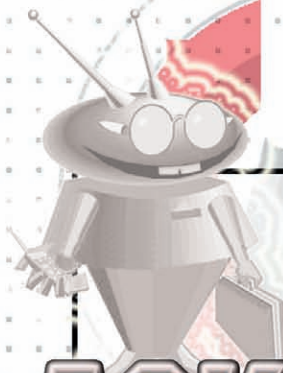
FREE LASER STENCIL WITH ALL PROTOTYPE PCB ORDERS

Watch "ur" PCB[®]
Follow the production of your PCB in **REALTIME**

email : sales@pcb-pool.com
Toll Free USA : 1 877 390 8541
www.pcb-pool.com

Beta

LAYOUT



Our resident expert on all things robotic is merely an email away.
roboto@servomagazine.com

Tap into the sum of *all human knowledge* and get your questions answered here! From software algorithms to material selection, Mr. Roboto strives to meet you where you are — and what more would you expect from a complex service droid?

ASK MR. ROBOTO

by
Dennis Clark

As you are reading this, in the back of your mind you're thinking that spring is right around the corner. As I am writing it, I'm thinking about what Santa will bring me. Not quite the same thing, but still distracting. Speaking of distractions ... I have had some interesting questions lately. I'm going to clear the table here at the end of the year and do my best to answer them, from the ones I thought were simple to the ones that were obscure. An interesting lot this has been!

It has been my experience that using web search engines can be a frustrating attempt to re-create the thought processes of the folks that post their information for us to find. Did the author call this a "circuit," a "board," or a "schematic?" Did she use "RC," "R/C," "Radio Controlled," or "Remote Control?" With that thought in mind, I went ahead and answered some of these questions for those out there who also might give up on the "cloud."

Q I have Futaba 8FG remote controls that I use on aircraft. How do I connect a remote control so the plane can then be controlled from a PC? Signals out of what? Thanks.

— Hananto

A Hanato, this is not so difficult as many would think. The manufacturers do not talk about how to do this, but the hacker community is very alive and very in tune with connecting transmitters to flight simulators, and creating cables for what the R/C aircraft community calls "buddy boxes" which allow a trainer to be connected to a pupil's transmitter so an instructor can take over when needed. However, this is a great way to void your warranty, so read on if you don't mind risking your radio!

Before you proceed, please note that I have *not* done this experiment with any of my equipment, so you will need to experiment with my suggestions to get your transmitter to be in "student mode" and accept signals from your computer. I do not recommend that you try to generate the

signals on your PC; it simply can't deal with the close timing required. I suggest that you instead use an embedded processor — one of many I have written about in the past — and have a serial port connection to your embedded controller board, and use your PC program to tell the embedded board what signals to send to your transmitter. **Fair Warning: It is not a simple project to control a model aircraft from your PC. You will be putting in a lot of time on this project.** If this hasn't frightened you away from this project, then read on.

On the back of your Futaba 8FG, you'll find a connector jack that looks like **Figure 1**; this is the "trainer cord" jack. This connector will have the signal inputs and outputs that you are interested in.

I recommend that you just buy a Futaba trainer cable from your local hobby store or on-line merchant rather than make your own. It will attach more securely. The **LXBEK7**

Figure 1. Trainer jack.

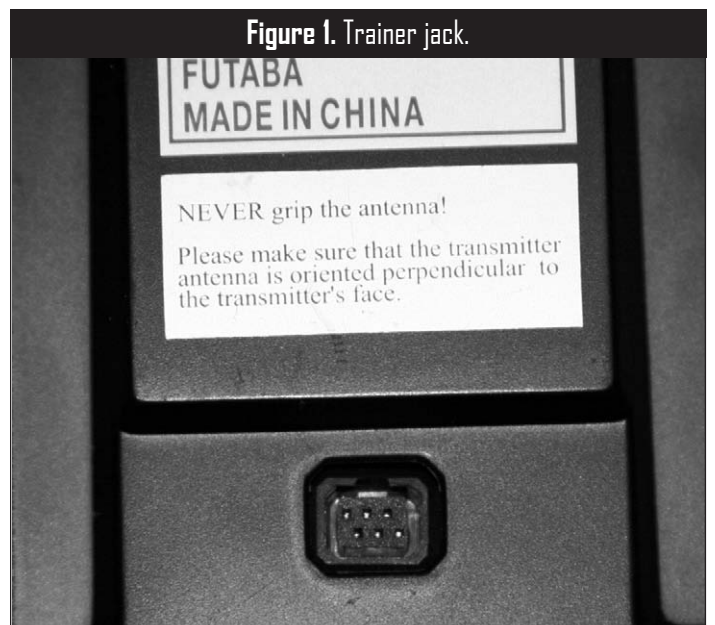


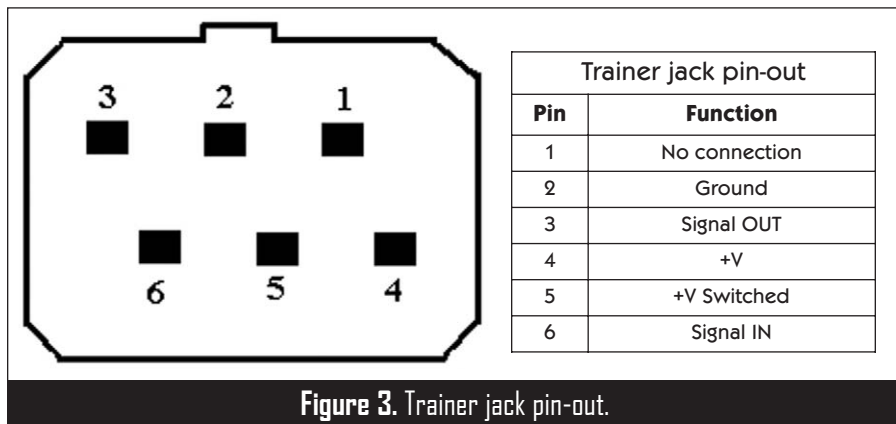


Figure 2. Trainer cord connector.

"Futaba trainer cord, micro to micro" cable from www.towerhobbies.com is a workable solution. The ends of this cable look like **Figure 2**.

This pin-out for the connector is available in a number of places on the Web. **Figure 3** shows a graphic of the trainer jack as seen from the back of the Futaba transmitter.

You are interested in pin 6 "Signal IN;" this is where you will be injecting your signal to control your airplane. At the minimum, you will need pin 2 "Ground" and pin 6. There is some discussion that if you connect pins 4 and 5 that the radio will stay on, but the RF (radio transmitter) will turn off. Sometimes you will need to send power down the shorted pins 4 and 5 at the same level you measure at pin 4 when your radio is on to get into this mode. Regardless, you don't want this so don't short those pins. Just turn your



radio on normally. Using an oscilloscope, look at the signal coming out of pin 3 to find out what your signal voltage levels are. You will want to inject the same voltage logic into pin 6 from your controller board.

What about that signal? You have no doubt seen any number of articles that describe the formation of a hobby servo control signal. The PPM signal that you will see coming out of pin 3 will look very similar to **Figure 4**, but different. This signal comes from my Futaba six-channel radio. These pulses are the various channels from 1 to 8 (yeah, 8, go figure) that I can control. Their timings represent the pulse values that will be sent to the hobby servos at the receiver after they are encoded in the radio transmission. Something that is not obvious is that these pulses are generated with a *negative* voltage with respect to ground. Remember that, when you generate your own signals. Also, the output voltage is about -4.2V. Play with your transmitter sticks and see how that changes the pulse timings. This will be essential for you to understand to create your own signals.

Now you know how to control your R/C aircraft from your PC (through your embedded controller board and trainer cable). This is just the tip of the iceberg, though! You will need to derive a program for your PC to intelligently control your aircraft and get heading, speed, and orientation data back from your plane. It sounds like a fun project. Please let us all know how it turns out!

Q - Can you suggest a supplier for encoders to be retrofit to stepper motors (NEMA 34)? Thanks.

— dk

A - There are a number of suppliers, but you would have to question their sales staff to be sure their mounting bolt patterns are compatible. Here are a few suggestions:

www.applied-motion.com or www.mclennan.co.uk

These companies seem to want to sell stepper/encoder combinations, but also have encoder add-ons. It isn't clear if they can be retrofitted.

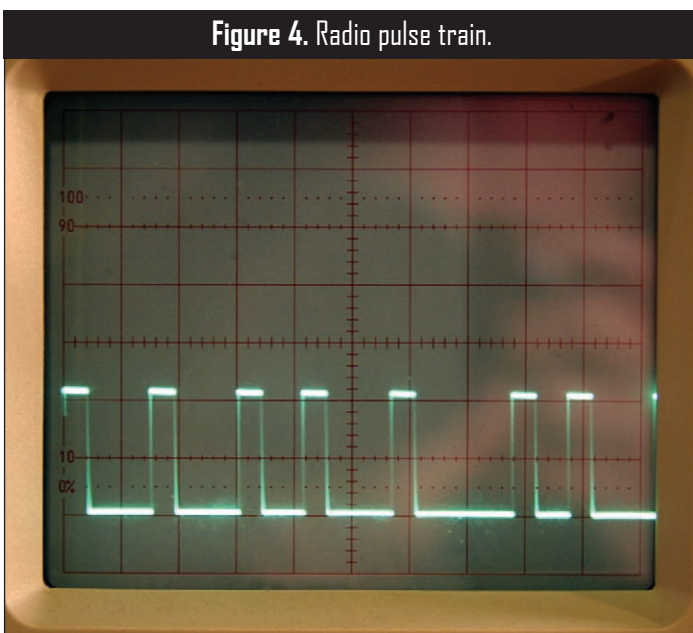


Figure 4. Radio pulse train.

www.quantumdev.com

The most promising company that I found (in my opinion) that has a variety of mountable encoders is this one:
Good luck!

Here's something a bit obscure. I researched this for a couple of hours and found that there is a LOT more out there on the 'net about hydraulic valves and cylinders than I ever imagined!

Q - My subject is micro-hydraulic components.
My questions are:

1) Is anybody looking at making a low-cost copy of the Moog E025 micro-hydraulic servo valve?

2) Is anybody looking at making double acting hydraulic cylinders smaller than the Hoerbiger LB model?

— William

A - I found little info about this servo valve. However, I did find the Atos E025 servo valve. Considering the uses for these components, however, I don't think that the cost will be less (www.atos.com). I was unable to get a price for what I think is the equivalent servo valve. The Hoerbiger LB cylinders are niche market (medical typically) and your requirements were a little vague. There are a variety of manufacturers and the LB series comprise a line from small to fairly hefty. In short, maybe. Not enough info.

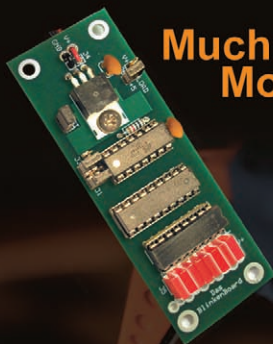
We've come to the end of another Mr. Roboto and as always, if you have a question, drop me a line at roboto@servomagazine.com and I'll be happy to try to answer it! Have fun and keep building robots!

SV

Das Blinkenboard

Not just for
blinken LEDs.

Much Much
More!



Complete kits
available @

<http://store.nutsvolts.com>

&

<http://store.servomagazine.com>

Firgelli
www.firgelli.com

LINEAR SERVOS



L12 -R Linear Servo

Direct replacement for regular rotary servos
Standard 3 wire connectors
Compatible with most R/C receivers
1-2ms PWM control signal, 6v power
1", 2" & 4" strokes
3 - 10 lbs. force ranges
1/4" - 1" per second speed ranges
Options include Limit Switches and Position Feedback

L12-NXT Linear Servo

Designed for LEGO Mindstorms NXT[®]
Plugs directly into your NXT Brick
NXT-G Block available for download
Can be used with Technic and PF
Max Speed. 1/2" per sec.
Pushes up to 5lbs.
2" & 4" strokes



PQ12 Linear Actuator

Miniature Linear Motion Devices
6 or 12 Volts, 3/4" Stroke
Up to 5 lbs force.
Integrated position feedback or
Limit switches at end of stroke
External position control avail.



Available Now At
www.firgelli.com

Robotics Showcase

**ALL
ELECTRONICS**
CORPORATION

THOUSANDS OF ELECTRONIC
PARTS AND SUPPLIES

VISIT OUR ONLINE STORE AT
www.allelectronics.com

WALL TRANSFORMERS, ALARMS,
FUSES, CABLE TIES, RELAYS, OPTO
ELECTRONICS, KNOBS, VIDEO
ACCESSORIES, SIRENS, SOLDER
ACCESSORIES, MOTORS, DIODES,
HEAT SINKS, CAPACITORS, CHOKES,
TOOLS, FASTENERS, TERMINAL
STRIPS, CRIMP CONNECTORS,
L.E.D.S., DISPLAYS, FANS, BREAD-
BOARDS, RESISTORS, SOLAR CELLS,
BUZZERS, BATTERIES, MAGNETS,
CAMERAS, DC-DC CONVERTERS,
HEADPHONES, LAMPS, PANEL
METERS, SWITCHES, SPEAKERS,
PELTIER DEVICES, and much more....

ORDER TOLL FREE
1-800-826-5432

Ask for our FREE 96 page catalog

Esduino12

- 9S12C 16-bit microcontroller in Arduino form-factor!
- 32K or 128K Flash
- optional USB interface
- low-cost Xbee plug-in option

Use with any
Arduino shield!



From
\$39

Easy object-based
Programming with nqBASIC!
Advanced programming in C
with CodeWarrior



Four new proto shields!

TechnologicalArts.com

EVENTS

Calendar

ROBOTS.NET

Send updates, new listings, corrections, complaints, and suggestions to: steve@ncc.com or FAX 972-404-0269

Know of any robot competitions I've missed? Is your local school or robot group planning a contest? Send an email to steve@ncc.com and tell me about it. Be sure to include the date and location of your contest. If you have a website with contest info, send along the URL as well, so we can tell everyone else about it.

For last-minute updates and changes, you can always find the most recent version of the Robot Competition FAQ at Robots.net: <http://robots.net/rcfaq.html>

— R. Steven Rainwater

FEBRUARY

- 2-5** **Kurukshetra**
Guindy, Chennai, India
 Robotics events this year include K!onstructor, Khimera, Pandemonium, XCEED, and Designer Quest.
www.kurukshetra.org.in
- 17-20** **Pragyan**
NIT, Trichy, India
 Events include Micromouse, Fix the Android, PIP Bots, and Crop Circles.
www.pragyan.org
- 17-20** **Techkriti RoboGames**
IIT, Kanpur, Uttar Pradesh, India
 This year's events include Reconnaissance, Robot's Got Talent, Trip to the Future, and FIFA 2050.
www.techkriti.org/#/competitions/robogames

MARCH

- 6-10** **APEC 2011 Micromouse Contest**
*Fort Worth Convention Center
 Ft. Worth, TX*
 High-speed, tiny robots solve a maze as fast as possible. The APEC conference runs March 6 through 10. The Micromouse contest will be on March 7th at 8 pm.
www.apec-conf.org
- 11-12** **AMD Jerry Sanders Creative Design Contest**
University of Illinois at Urbana-Champaign, IL

Robots must identify and collect colored Nerf balls.
<http://dc.cen.uiuc.edu>

- 12-13** **RobotChallenge**
Vienna, Austria
 Events include Micro Sumo, Mini Sumo, Standard Sumo, Parallel Slalom, and Slalom Enhanced.
www.robotchallenge.org
- 19** **Manitoba Robot Games**
Tec Voc High School, Winnipeg, Manitoba, Canada
 Lots of events including Standard Sumo, Mini Sumo, Mini-Tractor Pull, Super Scramble, Line Following, and Robo-Critters.
www.scmb.mb.ca

APRIL

- 9-10** **Trinity College Fire Fighting Home Robot Contest**
Trinity College, Hartford, CT
 The goal of this contest is to produce a computer-controlled robot capable of navigating through a mock house, locating a fire, and extinguishing the flames. The robot with the shortest time wins.
www.trincoll.edu/events/robot
- 10** **Robotics Innovations Competition and Conference**
Woburn, MA
 This event includes robot mobility through unconventional means.
<http://ricc.wpi.edu>
- 14-16** **VEX Robotics World Championship**
Kissimmee, FL
 This event includes high school and university VEX contests.
www.vexrobotics.com/competition
- 15-17** **RoboGames**
Ft. Mason's Festival Pavillion, San Francisco, CA
 This event includes FIRA, BEAM, MINDSTORMS, and Combat.
www.robogames.net



Now Serving Europe with Optimized Logistics

- Currency in EURO
- 3 day shipping to 70% of the territory
- Service in French and English
- Competitive shipping rates
- Huge product selection

One Global Door for Manufacturers

VISIT: **www.RobotShop.com** Shipping Worldwide

Robotics at your service! TM

NEW PRODUCTS

MISCELLANEOUS

Gripper and Pillow Blocks

Lynxmotion has been expanding their robotic offerings with a vacuum-powered arm gripper (VAC-KT). This arm attachment utilizes an inexpensive syringe and servo as the vacuum source. The vacuum generated is sufficient to hold a four ounce object for as long as 30 minutes. This assembly is a drop-in replacement for a normal gripper on any of their robotic arms, and can be assembled and installed with common hand tools. Made from high quality laser cut plastic and custom injection molded parts, this unique yet simple device will provide long lasting performance. For use in the classroom, science fair, mad scientist lab, or just for fun, the VAC-KT will make it possible to lift and manipulate small smooth objects with ease. It's priced at \$44.95 and comes with everything needed to upgrade an existing arm.

Due to the popularity of their 6 mm pillow blocks, Lynxmotion has expanded the range. They now stock 6 mm, 3 mm, 1/4", and 1/8" versions. These pillow blocks can be the foundation for many small projects that require supporting an axle in one or more places. The heavy duty shielded bearings are rated for 50K RPM. The pillow blocks share common mounting hole spacing and have the same axis height for ease of design. They're made from 6061-T6 aircraft grade aluminum, and priced at \$8.95 each.

For further information, please contact:

Lynxmotion

Website: www.lynxmotion.com



GEARBOXES

New Additions to Gearboxes

BaneBots announces new additions to their highly successful P60 family of gearboxes. In addition to the stock gearboxes, literally thousands of custom configurations will be available starting in January. Despite



their compact size, these gearboxes pack a lot into a small package. P60 gearboxes have been applied in an array of settings, including the educational, hobby,

commercial, and industrial markets.

With a new total of 34 ratios — ranging from 3:1 to 672:1 — the gearboxes can be customized in a number of ways. Many of the new options are a result of customer feedback and requests. The P60 shafts will now be available in 3/8" in addition to the original 1/2" diameter, and a hex shaft has also been added to the line-up. Short, standard, and long shafts give users more flexibility in design.

New, multiple mounting patterns add to the diversity of applications and ease of use. BaneBots also has increased the range of motor support to include industry standard RS-380, RS-390, RS-395, RS-540, RS-550, and RS-775 sized motors, as well as popular hobby motors

such as Speed 400 and rock crawler motors. Motors can be purchased either mounted or unmounted. Steel ring gear options give extra strength for those with more demanding operations, and the P60 gearboxes can be ordered greased or ungreased. In the near future, additional colors will be available for a customized look, as well. Another aspect of the P60 gearboxes is that they can be reconfigured to meet a customer's changing needs as a project evolves. If you own one P60, it is like owning several. BaneBots has all the gearbox parts available separately so that the ratios can be changed without purchasing a whole new gearbox.

The P60 has always been designed, manufactured, and assembled in the US at their Colorado facility. Components are CNC machined for tight tolerances and contain cold rolled steel gears and hardened 4140 steel carrier plates.

For further information, please contact:

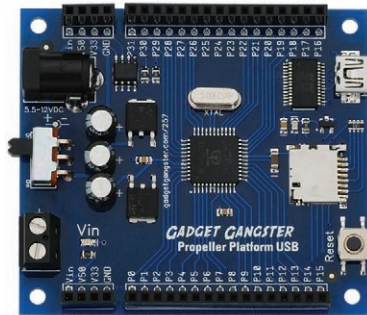
BaneBots

Website: www.banebots.com

DEVELOPMENT TOOLS

Propeller Platform USB

Combining a 64 KB EEPROM, 5 MHz removable crystal, 1.5A power regulation, USB, and a microSD card slot on a compact, breadboard and



protoboard friendly module, the Propeller Platform USB from Parallax is an easy-to-use development tool for the multicore Propeller microcontroller. All 32 Propeller I/O are available via pin sockets, along with 5V and 3.3V regulated power.

Features include:

80 MHz eight-core Propeller P8X32A with removable 5 MHz crystal; 64 KB EEPROM for long-term program and data storage; 5V and 3.3V 1.5A ultra LDO voltage regulators that accept 5.5V minimum power input; USB-to-serial interface for loading and communication; 2.1 mm center-positive barrel jack and screw terminal power connections; 2.8" x 2.5" footprint with pin sockets to add additional Platform modules or connect to a breadboard; and a microSD card slot connected to P0..P3

Retail price is \$49.99.

For further information, please contact:

Parallax

Website: www.parallax.com

mikroMMB for PIC18FJ board

The mikroMMB board from MikroElektronika provides a compact high-quality multimedia development platform

for PIC18 devices. It has numerous onboard modules that allow you to write multimedia applications of high complexity. It can be used for both development or as a final product.

For programming and debugging, an onboard PIC18FJ can be utilized by accessing the external programmer (18FJprog with mikroLCD support).

All microcontroller pins are available through breakout pads, so they can easily connect other peripherals with your board.

A large 320x240 TFT color display with touch screen and stereo MP3 Codec chip offer true power to build full multimedia applications.

This platform is excellent for beginners since writing code for the PIC18 will be simple and there is up to 12 MIPS operation and full speed USB 2.0 support.

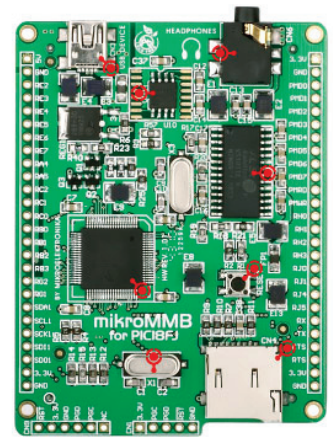
The board is carefully designed and is very compact and handy. Highest quality components are used, as well as a four-layer PCB to achieve maximum efficiency.

Users will be able to save pictures, sounds, and other media files on a microSD memory card and eight Mbit serial Flash memory.

For further information, please contact:

MikroElektronika

Website: www.mikroe.com/eng/products/view/585/mikrommb-for-pic18fj-board



TOOLS

Measure-Up Screwdrivers

Wiha Tools is now offering new "Measure-Up"



screwdrivers which are designed to quickly measure small parts on the job. They have both inch and metric measuring scales making it easy to check drilled hole depths and diameters.

The screwdrivers are available in both slotted and Phillips tip styles. They feature a SoftFinish cushion grip ergonomic handle shape which allows for maximum torque and user comfort.

For further information, please contact:

Wiha Quality Tools

Website: www.wihatools.com

bots IN BRIEF



WHAT THE HAL?

ReWalk is UC Berkeley's motorized robotic device that is controlled by motion sensors that feel the user's movements and translates them to motorized joints. The device goes outside clothing and consists of a harness, crutches, and leg braces powered by a rechargeable battery.

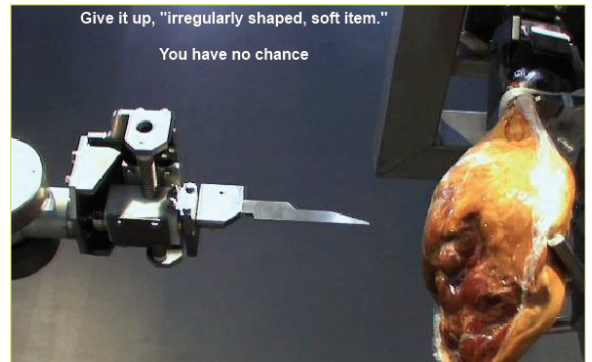
Argo Medical Tech — who built the ReWalk and has been testing it for the last several years — will soon be offering it to the public for ~\$100,000.

Other companies seem to have the same idea. Japan's Cyberdyne has produced an exoskeleton device called the Hybrid Assistive Limb (HAL). It anticipates the user's movements with sensors in a similar manner as ReWalk. HAL can be used to assist health and construction workers and firefighters, and is available for a rental fee of ~\$1,600-1,800 per month.

HAMMING IT UP

At the Fourth Robot Awards held at the National Museum of Emerging Science and Technology, top prize in the Small Business and Venture category went to HAMDAS-R — an automatic ham boning robot developed by Mayekawa Electric. Until now, ham boning has required the techniques of skilled workers, due to the variations in meat form and bone size. However, Mayekawa has successfully incorporated these techniques into the processing method, actions, and structure of HAMDAS-R, automating the task of boning ham.

Until now, boning 500 hams per hour required 20 people. Using HAMDAS-R, only 10 people are needed. The robot's consistent processing capability also makes production planning easier. Mayekawa intends to market HAMDAS-R worldwide, primarily in Europe and Japan.



PICKY PICKER

IAM-BRAIN (the Institute of Agricultural Machinery's Bio-oriented Technology Research Advancement Institution) has created a robot that can not only pick 60% of a strawberry crop, it can target the ripest and pick and cut in nine seconds. This berrybot's camera images the berries in 3D and uses algorithms to detect the redness. Farmers in Japan are now "field" testing the new technology.



CURIOUS GEORGE

Tin Woodsman flashback! Tony Sale built George — a six foot tall humanoid — from a crashed bomber in 1950. The former RAF officer recently brought his creation back to life with new batteries and some oil. You can see George at his new permanent location at the National Museum of Computing at Bletchley Park, UK that Sale helped found.



bots IN BRIEF

IN THE LINE OF DUTY

The Milwaukee County Sheriff Department's bomb squad used their service bot to help end a stolen Mercury SUV standoff on Interstate 94 recently when the inhabitants refused to cooperate. The robot's cameras showed that there were only two people in the vehicle and it broke a window before the cops tossed in some tear gas, forcing the two to come out.

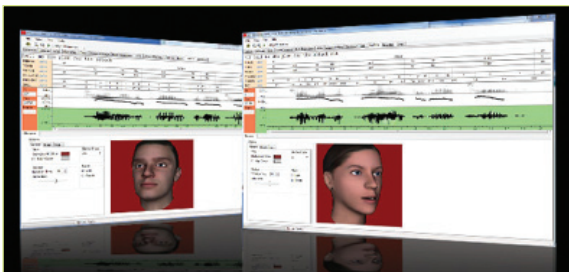
Incidentally, the driver, who was on parole and had previously been convicted of 14 crimes, had stolen the SUV from its owner, a man who picked them up in Illinois and trusted the couple because they gave him \$20. He left them to go into a convenience store and found his car gone when he came out.



RAPPIN' WITH ADAM

RMT Robotics® (www.rmtrobotics.com) achieved the first implementation of its ADAM RAP (Reactive Audio Playback) system at Otis Technology (www.otistec.com), a New York-based gun cleaning systems manufacturer and assembly operation. The new programmable sound system includes interactive voice messages and a mobile "vehicle in motion" jukebox for every mood and season. Together with the Otis operational group, RMT engineers customized the ADAM sound application to play "text to speech" messages, sound bites, or musical interludes that can be either actively or passively triggered in reaction to a variety of operational conditions and system inputs.

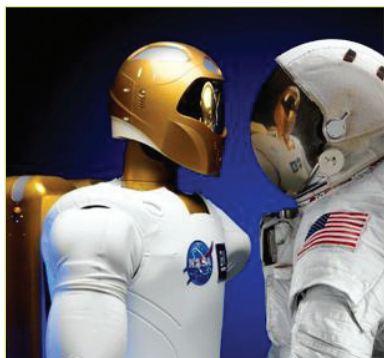
The ADAM RAP application is designed to play various sound bites in response to queues embedded in the control system to enhance the user friendliness of the overall system. The use of sound (either voice or familiar noises/melodies) apparently encourages the acceptance of ADAM by the human workforce and minimizes potential monotony in the workplace by reducing the sound redundancy that occurs with traditional beeper-based annunciations. ADAM robots have been working side by side with Otis employees since July 2009.



TALK TO ME

Google has reportedly purchased Phonetic Arts, a company that specializes in speech synthesis. They say that they will try to enhance computer robo-voices. The England-based company that they now own has produced voice creation PA studio, Composer which lets you create new lines of dialog from existing recordings, and Generator which is text turned to voice.

Cool tidbits herein provided by Evan Ackerman at www.botjunkie.com, www.robotsnob.com, www.plasticpals.com, and other places.



NO-GO FOR ROBONAUT

Don't send that robotic Bon Voyage message just yet. Robonaut 2 has to wait to make his trip to the final frontier. NASA is delaying the 39th and last flight so that engineers can study and run tests on the cracks in Discovery's external tank. Since they didn't make the December window, they are waiting for the next one to open in February.



SCAREDY RATS

When University of Washington researcher Jeansok Kim wanted to determine how animals dealt with fear when seeking food, he trained rats to retrieve pellets. Once they had mastered that, Kim added a LEGO Mindstorms Robogator that would snap at the rat when he left his shelter. To make a long story study short, the rats eventually learned that they could retrieve pellets 30" from the bot and avoided pellets that were only 10" away because that would put them in too close to Robogator.



X-37B TAKE TWO

The Air Force's X-37B landed successfully recently at Vandenberg AF Base in California after its seven month test run. The plane is about 29 ft. long, has a wingspan of 14 ft., and was built by Boeing's Phantom Works. Because the project is top secret, not many details are available concerning the unmanned vehicle's flight, but the results have apparently prompted the AF to order a second Orbital Test Vehicle that will be launched in the spring of 2011.



HUMANOID HAPPENINGS

The recent avalanche of new humanoid robots continued with a trio of robot reveals released at the end of 2010. PAL Robotics gave a presentation of their REEM-H2 service robot and their modular systems at the University of Sharjah in the United Arab Emirates last November. It looks like the robot's design has been

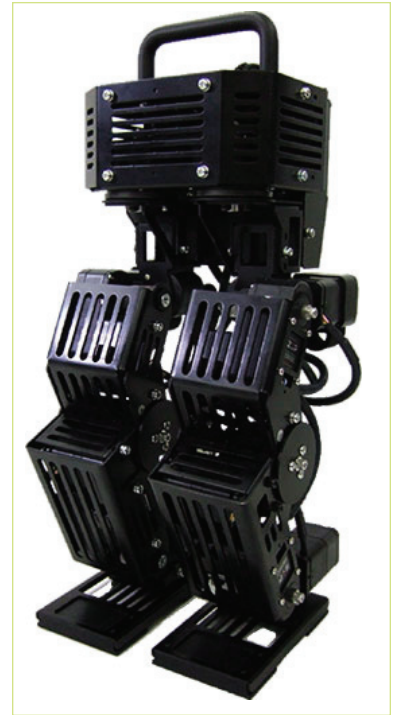
updated slightly from the original concept (the face has bigger eyes and it has proper hands). Virginia Tech RoMeLa and Robotis showed off the new DARwIn-OP (which was previously called the DARwIn-LC) humanoid robot kit in December at the Humanoids 2010 event in Nashville. Finally, UCSD's Machine Perception Lab and the Japanese robotics firm Kokoro Co., Ltd. teamed up to create Diego-san — a sophisticated humanoid modeled after a one-year-old child.



MYSTERY WALKER

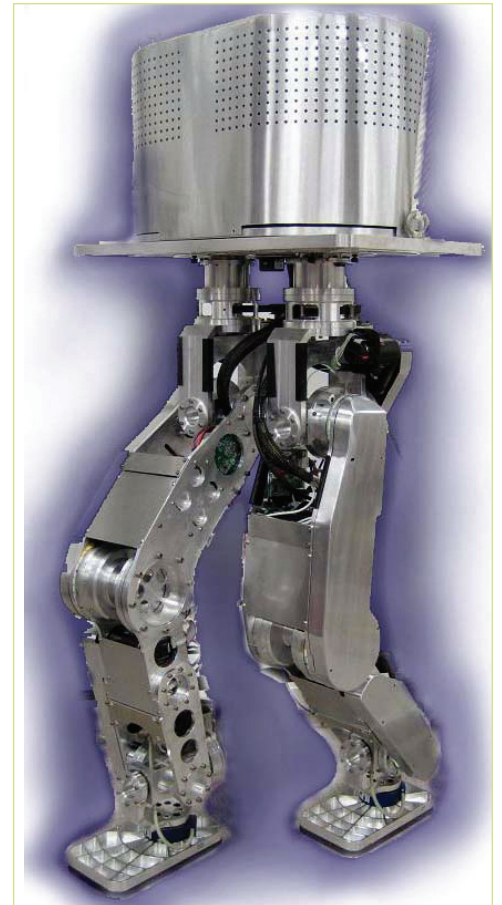
It appears that ZMP's President H. Taniguchi accidentally tweeted some unlisted YouTube videos recently of what appears to be his company's latest educational robot platform. The cat's not completely out of the bag yet, but we do know that this one has an upper body with arms (three degrees of freedom each), a single camera for a head, and a backpack large enough to hold a decent-sized processor.

Best guesses are that this new robot will cost around \$10,000 USD.



NO SERVICE

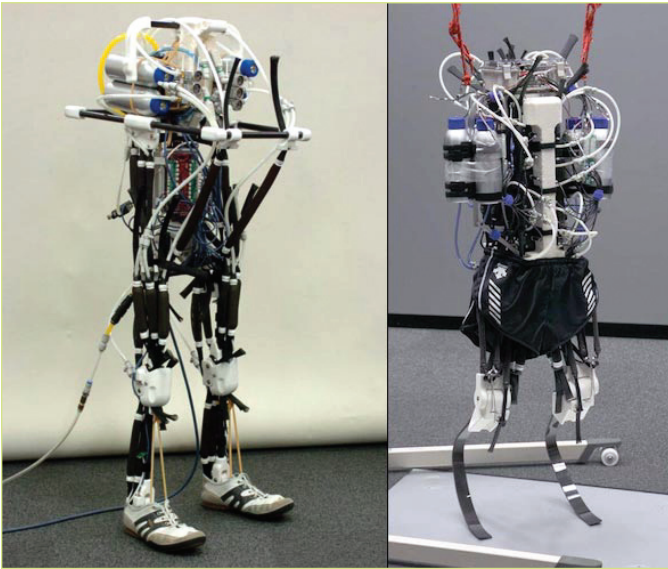
You can expect to pay a little extra when you go for a hot pot in China. The Dalu Robot Restaurant has robotic servers who cannot really serve the customers or pour beverages, but they tool around as the male waiters offer trays of food and the females dance between the tables. At a cost of \$6,000 a piece, the Shandong Dalu Science and Technology Company provided each bot with motion sensors so at least they can stop when a human or object is near.



MARI ME

One of the robots presented at the 2010 International Conference on Humanoid Robots was MARI-3 (pictured), built at Yokohama National University's Kawamura Lab back in 2006. It's a lower-body bipedal robot built out of aluminum alloy that is capable of walking and jumping up to 4 cm on one leg. It weighs 38 kg (83 lbs), has 13 degrees of freedom (two legs x6, with one yaw axis at the waist), and uses a gyro sensor, accelerometer, and six-axis force/torque sensors.

MARI-I — the lab's first biped — was built in 2000. It was 120 cm (4') tall and weighed 25 kg (55 lbs) with 14 degrees of freedom (two legs x6, two arms x1). It was able to walk at a rate of 1.4~2 km per second. Its successor MARI-2 followed in 2003. It was 130 cm (4'3") tall and weighed 70 kg (154 lbs) with 12 degrees of freedom (two legs x6). It had stereo CCD cameras in its head, allowing it to visually track objects and avoid obstacles.

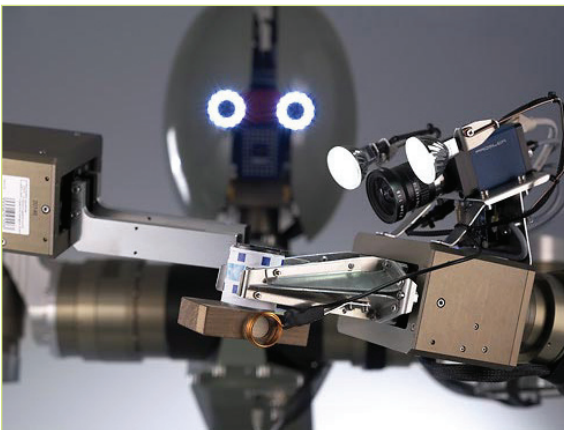
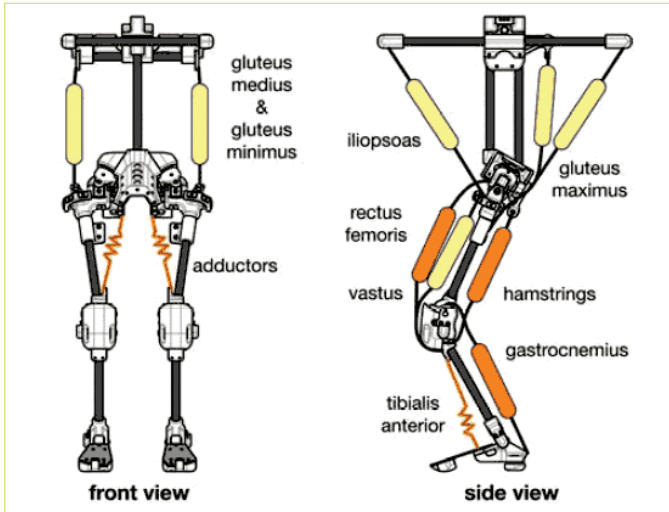


RUN, ROBOT, RUN

Traditional robots with geared motors have difficulty sustaining large, instantaneous forces over short periods of time. And, although researchers have tried to create bio-inspired robots with pneumatic or wire-driven actuators, few possess a biologically correct musculoskeletal structure. In order to study the natural capabilities of invertebrate animals, researchers at Tokyo University's ISI Lab (Ryuma Niiyama, Satoshi Nishikawa, and Yasuo Kuniyoshi) attempted to build a more accurate model with the Athlete Robot. They presented their work at the 2010 International Conference on Humanoid Robots in Nashville.

Unlike conventional systems that use on/off valves, this one uses proportional valves that can transform an electrical signal into a corresponding air flow. The McKibben artificial muscles have an extremely high power-to-weight ratio and are configured to match the human anatomy, including range of motion. For example, it is capable of jumping 50 cm into the air — which is remarkable for a robot of its size. Even when standing still it is more wobbly than its rigid cousins, but the sway is similar to that observed in humans. Thanks to its elastic properties (power and compressed air is supplied externally), the robot can land softly from a one meter drop — a big problem for conventional geared robots. The researchers found that they could also control the falling direction after ground impact by changing the stiffness of the joints. All told, the robot weighs about 10 kg (22 lbs) for all its 125 cm (4'1").

Earlier versions of the robot included pneumatic actuators for the lower leg, but they have opted for prosthetic blades similar to those worn by double amputees in the latest version. It has PID controllers at each joint, touch sensors in its feet, and an IMU on its torso for detecting the robot's orientation. Currently, it can only run for three to five steps (at 1.2 meters per second) before falling down, but the researchers are hopeful about improving its capabilities.



NICE PACKAGING

The Robotics Innovation Center at DFKI Bremen is showcasing their package-handling robot using a SemProM (Semantic Product Memory) system. An RFID tag with the package contents, weight, and other details can be read by the robot. Mr. SemProM was the prototype robot developed to test the system which will be replaced by the more finished AILA. It stands 175 cm (5'9") tall, and its two arms have seven degrees of freedom apiece. Stereo cameras help it locate objects while a camera on its left wrist gives it an even closer view during manipulation tasks. Interestingly, it has grippers capable of handling small objects (something AILA lacks), though it seems it was decided they were unnecessary for lifting boxes.

WORKERBOT TO THE RESCUE

Researchers at the Fraunhofer Institute for Production Systems and Design Technology IPK in Berlin have been working on a human-sized manufacturing robot thanks to the EU-funded PISA research project. The goal is to develop robots that can bring greater flexibility to industrial mass production to bolster European production lines. In Germany, manufacturers could lease the robots when necessary as bigger orders come in. The robot — called pi4-workerbot — is being brought to market thanks to pi4-Robotics.



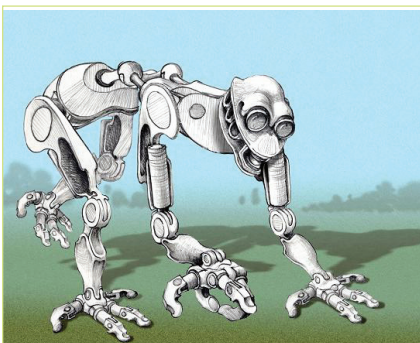
According to the datasheet, the robot can learn new tasks without needing to be programmed line by line. It has two seven degrees of freedom arms which allow it to transfer an object from one hand to another. This is

useful when examining the entire surface. Its manipulators boast “finger-tip sensitivity,” which means with the correct settings, the robot can hold an egg without cracking it. Its head is equipped with three cameras: a 3D camera observes its surroundings while the other two can be used for detailed inspection tasks. The face has a screen which displays various emotional expressions (it will appear happy if it’s busy working, but bored if it’s waiting for the next job).



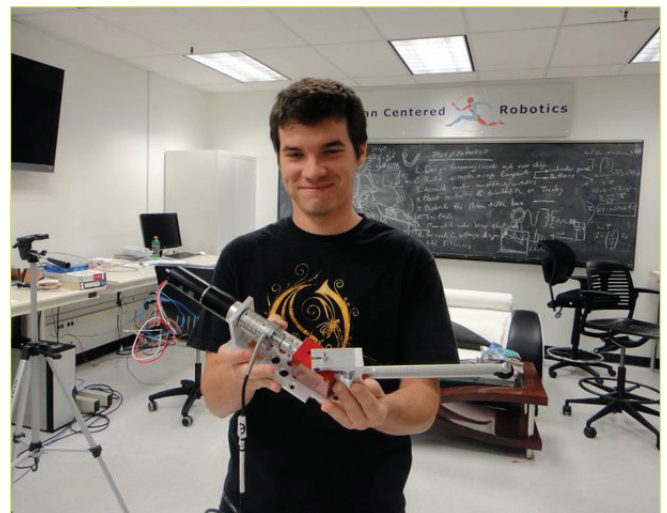
SCHOOL QUAD

Developed by students at the Human Centered Robotics HCR Laboratory at UT Austin, is a legged system called Ambuloid that uses a torque controlled active joint capable of delivering 10 Nm of torque. Ph.D. student Nick Paine is behind the concept and has developed a torque based feedback embedded controller/amplifier and embedded PC communication protocols using Linux Ubuntu with an RTAI kernel. These developments are



steps towards a full design of a compliant quadruped robot. M.S. Student Matt Gonzales has developed the mechanical design and finite element analysis to absorb the high impacts due to jumping.

To the left is an illustration of a concept of Ambuloid, a quadruped capable of advanced locomotion and dexterous manipulation.



COMBAT ZONE

Featured This Month:

Features

26 PARTS IS PARTS:
*DeWalt Drill Motor
Mounting Solution*
by Don Hebert

27 MANUFACTURING:
*Closed Loop
Lifter Solutions*
by Don Hebert

30 Potpourri
by Kevin M. Berry

Events

32 EVENT REPORT:
*Full Metal Carnage 2
Pushes Robot Combat
Over the Edge*
by Kevin M. Berry

33 Upcoming Events for
February - March 2011

PARTS IS PARTS

DeWalt Drill Motor Mounting Solution

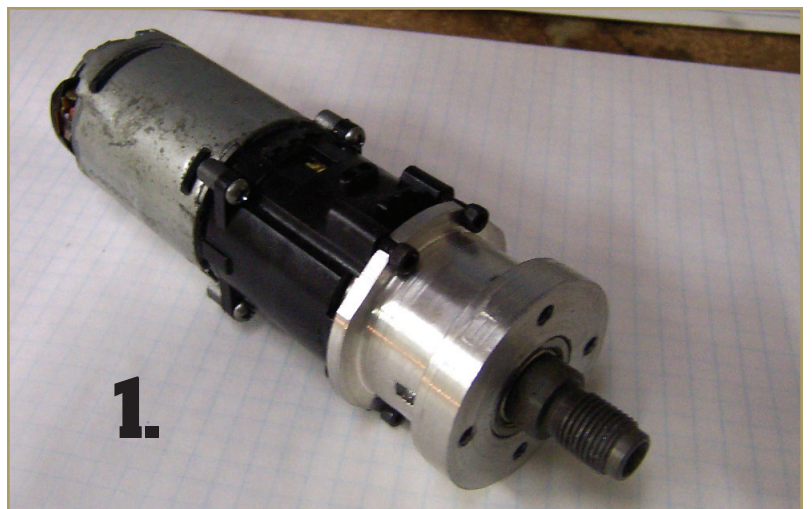
● by Don Hebert

The 18 volt DeWalt drill is used by many robots in the 30 pound and up category. A popular mounting is Team Delta's DeWalt Powerdrive kit, also available through **Robotmarketplace.com**. One difficulty with this mount is the output shaft must be supported externally with two bearings.

I made a mounting that uses the spindle shaft from the drill. It

has the 1/2-20 threaded nose where we attach a threaded adapter to mount the drive wheel. The spindle shaft has one bearing and the motor mount has an additional bushing near the transmission. This supports the wheel, transmits no side loads to the transmission, and helps center the floating final stage of the transmission. (See **Photo 1**).

The first version did not



support the motor, so it was done with additional framework in the robot. This was installed in my 30 pound combat robot "Trigger Happy." After two fights, the motor showed signs of coming loose from the transmission.

The second edition did a much better job supporting. Take a look at **Photos 2** and **3**. **SV**

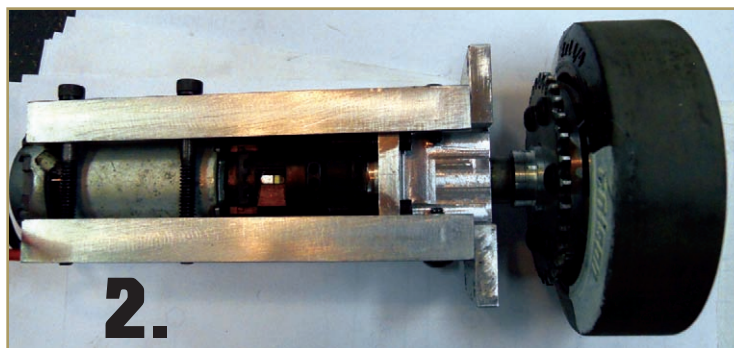
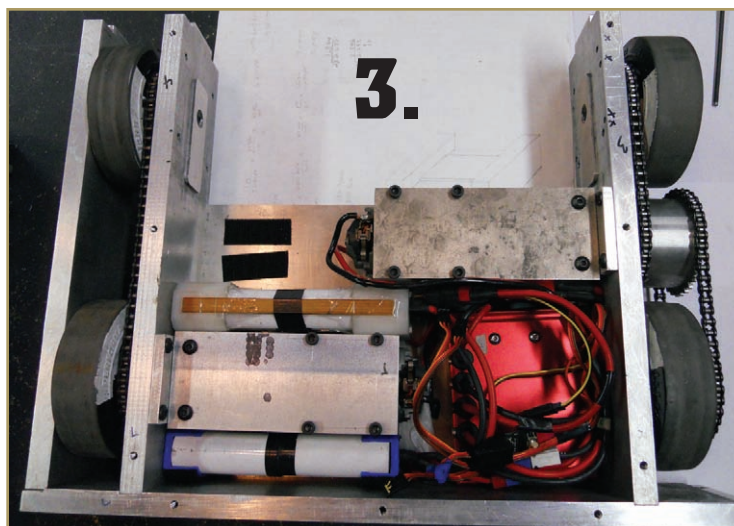


Photo 1: Version 1 motor mount/shaft support.

Photo 2: Version 2 supports the motor.

Photo 3: DeWalt 18V motors mounted in the frame of a 30 pound robot.



MANUFACTURING: Closed Loop Lifter Solutions

● by Don Hebert

I have worked with embedded microcontrollers for many years. So, when I began building combat robots in 2001, I was tempted to automate something. A lifter powered by an electrical motor seemed doable. I spent many hours solving mechanical and electronic issues and made a Victor883 stop automatically when reaching the top and bottom of the allowable lifter stroke. Those two PIC eight-pin microcontrollers I programmed in 2002 are still in use; I move them from one project to the next all the

time. Since then, I have tried other microcontroller products and off-the-shelf preprogrammed devices to see if they were satisfactory. I will describe the various solutions I have employed.

Basics of the Signals that Come From the Receiver

The radio receiver (Rx) outputs a pulse length proportional to the transmitter stick position. For a rudder, full left is one millisecond

(ms), middle is 1.5 ms, full right is 2 ms. The actual value can be modified by travel adjustments in the transmitter. The stick direction that is 1 ms can be changed by reversing that channel in the transmitter. The Victor 883 from Innovation First accepts this signal and can drive 60 amps. The lifter motor is a Harbor Freight 18V drill motor.

For the PIC, Arduino, and 9S12C, a socket for the microcontroller and a few wires were soldered onto a common

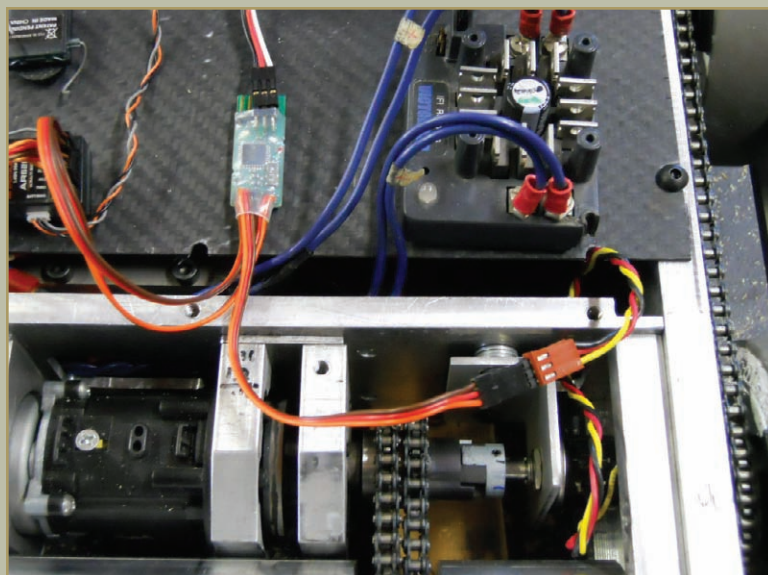


Photo 1. DigiMix3 connected to a Victor 883 and a potentiometer.

motor shaft. That connects to an analog input. The signal to the Victor 883 is driven so the position of the lifter is equal to the position of the transmitter stick.

With the R/C input connected to the gear channel, it can be made to go up fast and down slow by modifying the travel adjustment. This requires the least amount of attention during a fight. The switch is up or down. Connected to the rudder channel, it can be controlled as desired. This requires more splitting of your concentration while driving.

Programming is done in Assembly or C. Assembler requires very intimate knowledge of the details of the chip. It makes the fastest and (hopefully) best performance code, but is not for beginners. C is a higher level — like Basic in some ways — and is easier (for me at least). Performance may not be as good, but it gets the job done.

Description of the Various Microcontrollers I have Employed

1. Microchip PIC12C508. Made traverse with digital limits. Works well. It is an eight-pin chip that was programmed with the PICStart programmer and programmed in Assembly. It works well, but it is not easy to program for beginners to microcontrollers. If I were to continue with the PIC, I would use the PICit3 programming device that leaves the chip on the project board while programming and debugging. It is possible to program in C but I never got that far with it. Programming tools are not high priced, but they are not free.

2. Motorola 9S12C from Technological Arts. Made analog

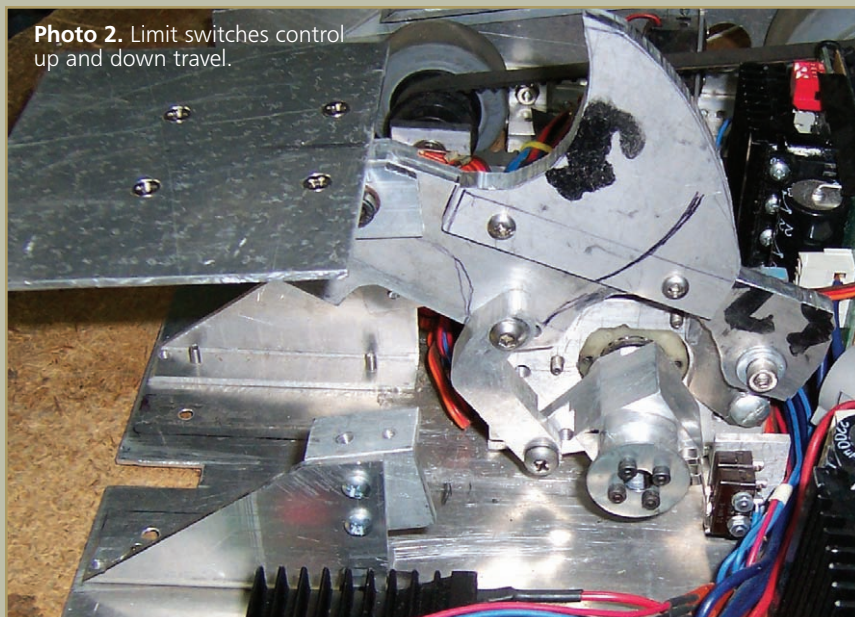


Photo 2. Limit switches control up and down travel.

prototype board. The control boards connect between the RX and the Victor 883. **Photo 1** shows a typical installation.

For the traverse with digital limits, the switches are connected to digital inputs for max up and max down limitation. The R/C command from the gear or rudder channel is used for input. If the command is positive and we are not on the up limit, the signal is retransmitted. When the up switch is tripped, a neutral pulse length is output stopping the motor even if the transmitter stick is still at full

deflection. The mechanical limit switches are mounted at the ends of travel. **Photo 2** shows limit switches operated by cams.

For traverse with analog limits, the potentiometer connects to the motor shaft. That connects to an analog input. Trim pots on the controller board are connected to two more analog inputs and they set the up and down position allowed for the motor position. **Photos 3** and **4** show the potentiometer installation.

The true closed loop variation also connects a potentiometer to the

closed loop. Size of a BASIC Stamp board. Programs in C using free CodeWarrior compiler. There is no USB or RS-232 on the chip. Programming requires chip to be connected to a development board, then placed on the final circuit card that will fit in the robot. The alternative in-circuit programming requires additional hardware, and I would buy that if I were to work with this chip more. I liked the compiler and the chip is powerful.

3. DigiMix3 from Firmtronic; made analog closed loop. This looks like a real small mixer with R/C cables for input and output. There is no soldering iron required for this one. It comes with three R/C input cables and two output headers (pins) for standard R/C connection. One of the inputs can be used for analog input. It has preprogrammed standard mixer functions that can be downloaded with the provided program adapter and software. It is also customizable; I made it a true closed loop. It is an Atmel AVR processor. The programming is C using the free Programmer's Notepad — a component of the AVR development toolkit.

4. Arduino Nano; made into a digital up/down limit and closed loop. Size of a BASIC Stamp board. Drives a Victor 883. Again, an AVR microcontroller. Programmed in C with the free Arduino compiler/downloader. The USB driver chips are on the Nano. I cut up a piece of prototype board, soldered R/C cables, and trim pots. In an hour, I was downloading and testing.

The next few are controllers with included motor drivers, replacing the Victor 883.

5. MotionMind from Solutions Cubed. It does not require programming. I configured it closed loop with the provided software. The motion of the stick is followed by the motor. Connected to the throttle, it will stay where you put it.

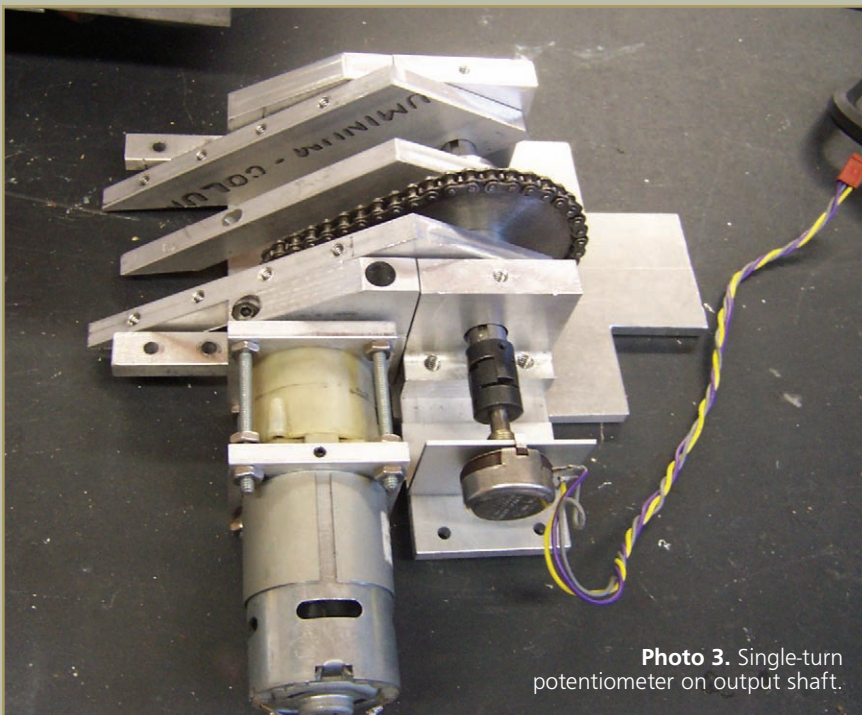


Photo 3. Single-turn potentiometer on output shaft.

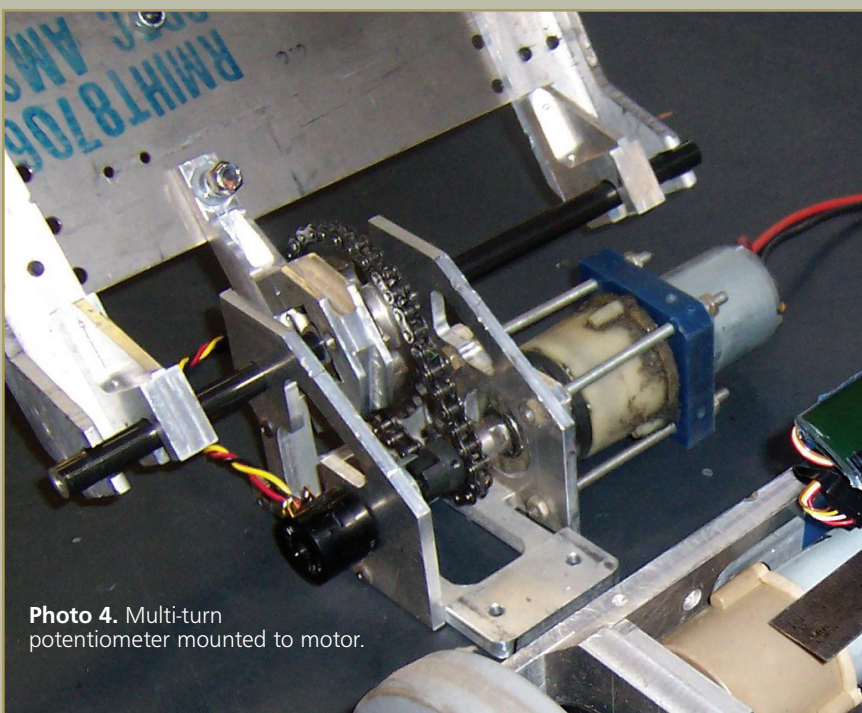


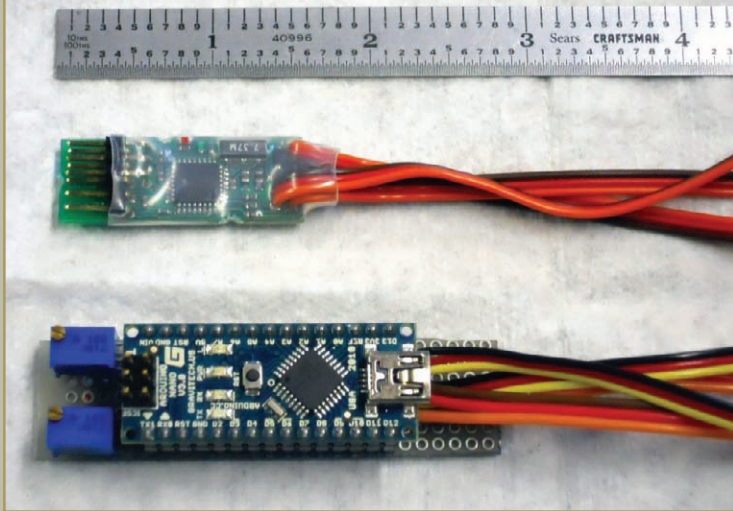
Photo 4. Multi-turn potentiometer mounted to motor.

It works very well but it is much smaller current capacity, compared to the Victor. I was not able to drive a HF 18V with it. Would be a good fit to a beetle or some other smaller project.

6. AX1500 from RobotEQ. Equal in capacity to a Victor when the two channels are slaved together.

Configured with supplied software to various programmed options; I picked closed loop. At power up, the lifter moves some before settling down. This makes passing safety a problem. I added a RCE210 relay to the motor output's connected gear channel. This way, failsafe and power-on is done with the gear

Photo 5. Top is DigiMix3, below is Arduino Nano with the hand-wired connection board underneath.



retracted and the motor is physically disconnected from the AX1500. Gear-on enables the lifter motion. Throttle channel controls position. While this worked well, it is very large compared to a Victor.

7. AX500 from RobotEQ. Smaller amp version of the AX1500. Same pros and cons.

The MotionMind and AX1500/500 do what they were designed for well, and have many functions that would be hard to equal. If their built-in features match your needs, they are an excellent off-the-shelf solution. However, customizing beyond what it already does is

not possible.

For an experienced programmer, the PIC products and Motorola 9S12C are good embedded microcontroller solutions. Very powerful, but not for the beginner. Other controllers that may have merit, but I have not used are the BASIC Stamp and Propeller by Parallax; and Atom, Atom Pro, and Atom Nano by Basic Micro.

The DigiMix3 is easy to use, and can do some things well but not everything. Price is \$50 and it is available from **Robotmarketplace.com**.

For the average weekend program warrior, the Arduino Nano is a good first choice. Even this small chip (the size of a BASIC Stamp) has a USB programming connector. The Arduino website links to various suppliers. I got mine for \$35 from Gravitech at <http://store.gravitech.us/arduino.html>.

I consider an oscilloscope mandatory to see what the inputs and outputs are doing. I use a Hobbylab USB scope that costs \$130.

I hope this helps if you want to tinker with closed loop lifter control or microcontrollers in general. The advances that have been made in the last 10 years are amazing. Microcontrollers are within reach of even the most budget-minded hobbyist. There are plenty of tutorials on line for whatever route you take. You can get started for \$50 or less. Add \$100-\$200 if you need a USB scope. Good luck! **SV**

POTPOURRI

● by Kevin M. Berry

Having received zero negative feedback on the initial version of Potpourri in December '10's issue, I've decided to do it again. (Of course, I didn't get any positive feedback either. I never do. You never write. You never call. You don't email me any more ... sorry! Midwest mom guilt-trip flashback.)

Here's some "take" from the forums. Hope you find some of it sweet smelling — like machine oil, magic smoke, and freshly cut titanium.

Lithium-Ion vs. Lithium-Polymer

A builder writes: "I'm looking into a new build and would like to use a higher density battery system besides the aging NiCad and NiMH cells I have been using. The Lithium-Ion or Lithium-Polymer batteries seem to be what I'm looking for. In my research, it seems the Li-Ion system is more robust than the Li-Poly, and doesn't require such care when

charging and discharging. I haven't found any information on Li-Ion packs requiring balancing like the Li-Poly packs do. I already



have a charger capable of charging both (Multiplex LN-5014).

Is the Li-Ion system more robust? Does it require balancing or other detailed maintenance? Besides temperatures and fire risk, what concerns should I have in combat?"

Wisdom of the Forums

The A123 Li-Ion cells are certainly a good choice, but they have their own care needs just like any other technology. Li-Poly is also a good choice, and again has its own set of needs. Some of it just depends on what you are building.

Always balance lithium-based cells if you can, no matter which exact chemistry. Even if the manufacturer tells you it doesn't need balancing, your cells will be happier if you do.

Some points to consider for A123 cells:

1) They are easily physically damaged. They are very light, so the metal casing is pretty thin, and they dent and bend more easily than other chemistries. Shock pad accordingly!

2) They will get out of balance. Not all cells are the same, and newer cells are much better than old. Try to balance-charge at events if you have time, but if you have to just bulk-charge, hope they stay close enough to not damage any of them. Understand you're taking the risk of destroying them if you do this.

3) Specs on some say 70A continuous and 120A peak. They will do over 200A on a dead short.

4) Do NOT trickle charge to balance like you would with NiMH or NiCD. The cells will just continue to climb in voltage and you will kill them.

The iMAX B8 charger came recommended as part of this string, to use in balancing Li-Ion cells. Apparently, the other advantage of A123 is they don't do the "swell up and burn like crazy" thing like

a Li-Poly can.

While these A123s are amazing, a little research shows they are — like all optimum solutions — on the pricey side. Buyers are looking at \$8 to \$14 per cell, so that's \$100++ for a 12 volt pack.

On CAD-ing

Lots of layouts are done on graph paper. Of course, it's much easier to revise on the computer. A good, free, easy 2D CAD program is called eMachineShop (www.emachineshop.com).

Another popular answer is Google Sketchup. It's free and intuitive, and the documentation is really good. The only downside is that it is hard to import and export other CAD formats. If you are a student, you can also get Autodesk Inventor for free sometimes. With small robots, CAD software is nice because you can print out cutting templates.

Sticking CF

"Does anyone have an opinion as to the best adhesive for carbon fiber like the kind Robot Marketplace sells? In addition to gluing the two rough sides together, I'm also looking for advice for gluing the rough side to the smooth shiny side of Texalium.

Then, there's UHMW. Are there any adhesives that work well with this stuff? I plan on using screws but I figure it doesn't hurt and it helps get the holes lined up.

And The Forum Says

"Shoe goo" or 'goop' — same thing — for the carbon fiber. And forget gluing anything to UHMW. I am pretty sure

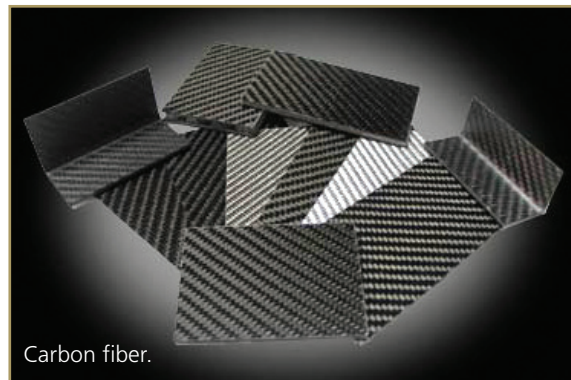


it is impossible."

"I'm unfamiliar with that Robot Marketplace product, but I'd *assume* that the matrix in the composite is an epoxy. You want your adhesive to be as chemically similar to your composite matrix as possible. Before bonding, you want to clean both surfaces thoroughly with a solvent, abrade both surfaces with *clean* sandpaper, remove all debris with a clean dry cloth (don't use a solvent again or you'll dissolve impurities and release agents right back into the surface!), let it dry thoroughly in a clean environment, then bond."

One responder noted: "The Velcro tape and double-sided mounting tape I've used stuck very well to UHMW. So, there are some adhesives that will stick well to UHMW."

And another: "I used shoe goo to stick it to aluminum without failure. I wire brush/roughed up and cleaned both surfaces. On an ant — when I was looking for strength — I used to use a Devcon plastic welder and had



Carbon fiber.

no problems. Go to **www.devcon.com/products/products.cfm?familyid=182** or it's available at Ace Hardware. I think a little flexibility is a good thing."

"I've heard that there's two things you can do to improve your

chances of bonding to slippery things like UHMW: 1) significant mechanical abrasion; and 2) flame treatment. It will roughen the surface and change the properties enough to enable bonding."

Regarding bonding RMP

carbonfiber: "Surface prep, bonding conditions, and mechanical design will play as big a part in whether the bond fails as which adhesive you use. Make sure the surfaces fit together well, provide the largest surface area you can, and clamp them well during curing. **SV**

EVENT REPORT: Full Metal Carnage 2 Pushes Robot Combat Over the Edge

● by Kevin M. Berry

The Queensland Robotics Sports Club teamed up with The Edge to present two full days of remote controlled carnage on November 27th and 28th. Thirty Ants, Beetles, and Feathers fought it out in a fully weaponed, hard fought series of battles. The event was held at The Edge — an offshoot of the Queensland State Library. The Edge is an arts and technology center geared at the younger crowd, and robot combat is a perfect fit for their mission.

One highlight was "Demon," from builder Glen Rose. The weapon — a three kilo vertical disk — spun at 6,500 RPM for early

fight. During the event, QRSC officials requested Glen to reduce the RPM, but it still won the Featherweight division against Nick Martin's horizontal spinner "Scissorhands."

Major media coverage from newspapers and the local ABC affiliate — coupled with lots of high speed video — made for a very well video'd event. Besides a LOT of YouTube postings, the QRSC posted some high speed footage at <http://qrsc.org.au/wikka.php?wakka=TheEdge2>. **SV**



RESULTS:

Antweight Division

- 1st - Scarlet
- 2nd - Stig
- 3rd - Fury Jr.

Beetleweight Division

- 1st - Polycarbide
- 2nd - Mini Flexo
- 3rd - Sacrificial Lamb

Featherweight Division

- 1st - Demon
- 2nd - Scissorhands

EVENTS

Upcoming Events February - March 2011

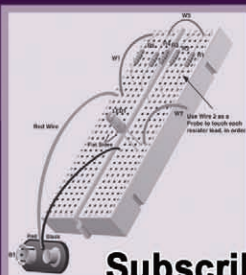


Motorama 2011 will be presented by the North East

Robotics Club, in Harrisburg, PA
on February 18 through 20, 2011.
www.nerc.us



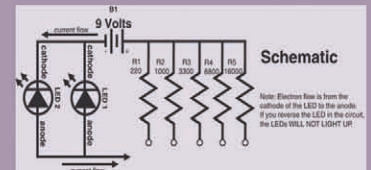
BB 2011 Nationals will be presented by BattleBots at the Coconut Grove Expo Center in Miami, FL on February 23rd through 27th.
www.battlebots.com. **SV**



FUNDAMENTALS For The Beginner

Need the Basics?

Follow along with our series of articles which includes easy to understand graphics. Starting in the May 2010 issue.



Subscribe Today! (with optional digital back issue viewing.)

Visit www.nutsvolts.com or call (800) 783-4624

Extreme Robot Speed Control!

Sidewinder

\$399

RC Control

- ◆ 14V - 50V - Dual 80A H-bridges - 150A+ Peak!
- ◆ Adjustable current limiting
- ◆ Temperature limiting
- ◆ Three R/C inputs - serial option
- ◆ Many mixing options - Flipped Bot Input
- ◆ Rugged extruded Aluminum case
- ◆ 4.25" x 3.23" x 1.1"

BotsIQ Favorite!



\$39.99

Scorpion Mini

- ◆ 2.5A (6A pk) H-bridge
- ◆ 5V - 20V
- ◆ 1.6" x .625" x 0.25"



\$159.99

Scorpion XXL

- ◆ Dual 20A H-bridge 45A Peak!
- ◆ 5V - 28V
- ◆ 2.7" x 1.6" x 0.75"



\$104.99
2+ price

Scorpion XL

- ◆ Dual 13A H-bridge
- ◆ 5V - 28V
- ◆ 2.7" x 1.6" x 0.5"

Dalf Motion Control System

- ◆ Closed-loop control of two motors
- ◆ Full PID position/velocity loop
- ◆ Trapezoidal path generator
- ◆ Windows GUI for all features
- ◆ Giant Servo Mode!
- ◆ C source for routines provided
- ◆ PIC18F6722 CPU

\$250



See www.embeddedelectronics.net

H-bridges: Use with Dalf or Stamp

NEW!

Magnum775

- ◆ planetary gearbox
- ◆ 20:1 ratio - 700 rpm
- ◆ RS-775 motor
- ◆ Nearly 700W!
- ◆ Build something - rule BotsIQ!

\$89



\$79

Simple-H

- ◆ 6-28V 25A!
- ◆ 2.25"x2.5"x0.5"
- ◆ 3 wire interface
- ◆ current & temp protection



www.robotpower.com

Phone: 253-843-2504 • sales@robotpower.com



We also do consulting!
Give us a call for a custom motor control to meet your exact needs



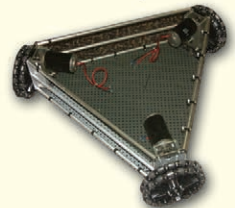
Inspiring Mobility

Specializing in Unique Wheels, Gearboxes, Aluminum Sprockets and Drive Bases



6" Aluminum Dualie Omni Wheel

Tri- Lambda Drive Base
Omni-directional drive system kit.



Aluminum Sprockets

8" Mecanum Wheel



AndyMark, Inc.
700 E. Firmin St., #114
Kokomo, IN 46902

765-868-4779

www.andymark.com

The Robotis OLLO Explorer

A Great Starter Robot Kit For Kids

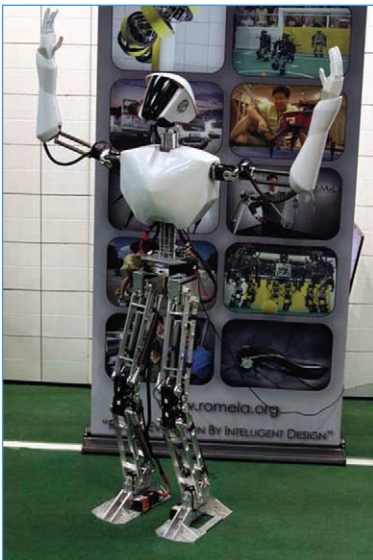


FIGURE 1. CHARLI-L built by Virginia Tech with Robotis Dynamixel actuators.

I have long admired the Robotis Bioloid line of robot kits and their very unique Dynamixel rotary actuator servos. Universities around the world have based much of their robotics curriculum upon the South Korean company's Robotis Bioloid and Dynamixel designs. Dennis Hong and his students at Virginia Tech built CHARLI-L (shown in **Figure 1**) using Robotis actuators. It is a small 54 inch human-sized walking humanoid that has been featured in many scientific magazines. The downside of the Robotis Bioloid line of products is their relatively high cost.

With that in mind, Robotis has developed a series of low cost robot kits for young kids to help them learn about and build robots. These introductory kits have garnered much praise at toy fairs around the world as they acquaint youngsters to the sciences of physics and mechanics in a creative and fun way. As Robotis states: "OLLO is science (great introduction to physics and mechanics), creativity, variety, entertainment, logic (the RoboPlus programming language can be used for all levels of robotics), power, safety (no soldering/simple tools), and design."

FIGURE 2.
The Robotis
Explorer Kit.



In April '10, RoboGames was held in San Mateo, CA. There were quite a few booths with displays that ranged from robot toys, robot accessories, electronics companies, robots, and robot kits. Among several that really drew attention, the Robotis OLLO booth seemed to draw the most kids and their parents. In the April '09 issue of *SERVO*, Bryce and Evan Woolley wrote a great article on the OLLO individual Bug Kits in their column, *Twin Tweaks*. These kits can be assembled into an intelligent robot for a shade under \$100 retail.

The simplest of these offerings are the Figure kits retailing for about \$20, and they are appropriate for ages 7+. These kits are not motorized. The Action kits have a single motor powered by an AA battery, cost about \$30, and are for ages 8+. The OLLO Starter kit includes 12 simple action robots and a nice 160 page color manual. The Bug kits just mentioned add an intelligent module, remote control, can follow a line path, cost about \$100 each, and are for ages 12 and above. The new OLLO Explorer Kit retails for \$149.90 and is the kit we'll discuss here.

The OLLO Explorer Kit

The Explorer kit shown in **Figure 2** is a complete construction set that allows a child to build one of 11 robots with detailed instructions, or construct a robot of their own design. The kit contains a CM-100 microcontroller (**Figure 3a**) with built-in sound and light sensors, two gearmotors to provide motion (**Figure 3b**), computer interfacing cables for programming, a software CD-ROM, three very comprehensive (yet child-appropriate) instruction manuals, and numerous fastener rivets to fasten the many hole-filled plates used in the construction of all robots. **Figure 3c** shows the battery box that connects the power to the controller.

It is the rivets shown in **Figure 4** that make the OLLO kits unique and different from LEGO kits. They remind me of the rivets used in the Androbot TOPOs from the mid-80's in that they have a center pin that locks the rivet in a hole. Robotis furnishes a unique tool shown in **Figure 5** to insert the rivets/pin and also to remove them. This tool is necessary in the assembly process as the rivets and pins are a bit small for most people to handle — especially kids. This all-purpose gadget is really a plus.

James Chi and Jinwook Kim of Robotis offered one of the OLLO Explorer kits for evaluation. I wanted an unbiased opinion about the new set so I decided to have a young neighbor boy named Luc Bennet actually do the construction of the robots since he was not related to me. (He got to keep the kit as a thank you for helping with this article.) Luc has long been interested in robots and has spent many hours with the LEGO Mindstorms series in a number of robotics classes, and it was these types of kits that piqued his interest. Luc is 9 years old, enjoys soccer, baseball, math, and plays a mean game of chess. In fact he's competed at state championships. He wants to be an engineer, though working with animals and being a 'computer spy' rank high, as well.

Luc Uses the Explorer Kit

Luc had attended several robot camps at the Oregon Museum of Science and Industry, OMSI and thoroughly enjoyed using the powerful NXT Intelligent Brick to power some of the camp's LEGO robot creations. LEGO has a lower priced educational set for kids called WeDo Robotics that allows the construction of 12 different projects for \$130 retail. These kits rely on a USB interconnection to a laptop or desktop computer for intelligence and power.

My goal was to have Luc select the robot that he wanted to create from the kit and build it on his own without me or his parents being involved. I wanted him to form his own opinions of the kit, the instruction manuals, and the difficulty level. I simply

handed him the kit and told him to "Go for it." He did.

Presented with a choice of 11 robot designs to select from,

Luc's first robot build was the Howitzer — a robotic catapult of sorts. Luc's start of the construction is shown in **Figure 6**. The next day, Luc called me over to his house to show me the robot (see **Figure 7**). Mechanically it was all together, but he was having a little trouble with the programming. I was tempted to sit down and work with Luc on this, but decided to have him try to figure it out on his own. In the end, he looked up examples of programs on the Robotis website to get a feel for the commands.

His mother helped him a bit and he got the hang of it in pretty good order, and completed the RoboPlus Task programming on his own. The included CD-ROM has a number of downloadable programs (available for Windows only) and the source code for the more advanced behaviors and projects described in the third manual. The nice thing about the RoboPlus programming environment is it is the base system used in the Bioloid kits and should allow a kid to comfortably progress to these more advanced kits and systems at a later time.

Luc's Opinion of the OLLO Explorer Kit

Next, Luc built an OLLO robot of his own design and

FIGURE 3a.



FIGURE 3b.

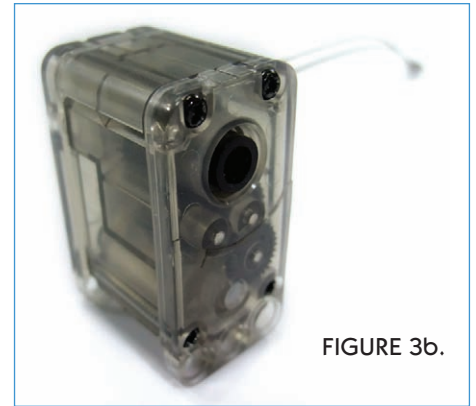


FIGURE 3c.

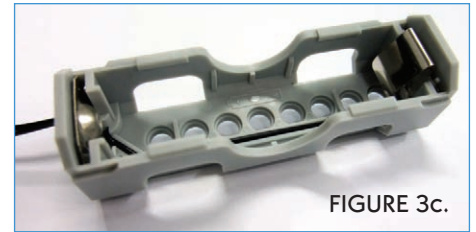


FIGURE 4.
The OLLO
rivet/pin
arrangement.



FIGURE 5.
The OLLO
rivet/pin
tool.





FIGURE 6. Luc begins his build.



FIGURE 8. Tho OLLO mouse senses Luc's hand.

later constructed a 'mouse' shown in **Figure 8**. He then built the car design and had a lot of fun just building robot-like things with all the parts. I debriefed him on his

experiences with the Explorer kit, knowing that he had a bit of a preference for the LEGO robot construction kits that he had used previously. I wanted to know how he compared the two different brands. One thing that impressed him about the OLLO Explorer kit is the robots don't accidentally come apart as easily as the LEGO kits, however, he noted that the LEGOs snap together easier.

He also liked the idea of being able to design and run his own programs with the OLLO kits. He admits there are more of the LEGO robots around and he is more familiar with them, but the lower cost of the OLLO kit was something that he felt would allow more people to buy it for their kids.

Luc enjoyed making things with the Robotis OLLO Explorer kit for several months — something that interested parents should consider. I should note that there are many very tiny parts associated with the kits and these should be kept away from smaller children. The quality of the manuals and included projects are excellent in helping kids (and adults) from the ages of 8 and up to thoroughly understand the physics, mechanics, and control systems required to develop numerous electro-mechanical and robotic projects.

I can easily see why educators at many fairs and expositions have praised this new Robotis line of robot kits. I personally enjoyed building a few of the projects, myself. Go to **robotis.com** or call some of the advertisers here in *SERVO* for further information. **SV**

www.servomagazine.com/index.php?/magazine/article/february2011_Carroll



FIGURE 7. Luc's Howitzer.

Intelligent Vacuum Cleaner Systems

www.servomagazine.com/index.php?/magazine/article/february2011_Blankenship

by John Blankenship and Samuel Mishal

Creating a simulation of a robotic vacuum cleaner makes it easy to explore more advanced behaviors than those found in commercial products. Imagine your robot vacuum automatically scurrying to clean up when you accidentally spill some popcorn.

In the November '10 *SERVO*, Editor Bryan Bergeron challenged readers to find economically feasible solutions for making home vacuuming robots more intelligent. Current systems generally perform their work by randomly roaming around the room, hoping to accidentally encounter bits of litter. One of Bryan's goals was to have a system that could detect accidents (such as spilled popcorn) and immediately move to the affected region to clean the area quickly and efficiently.

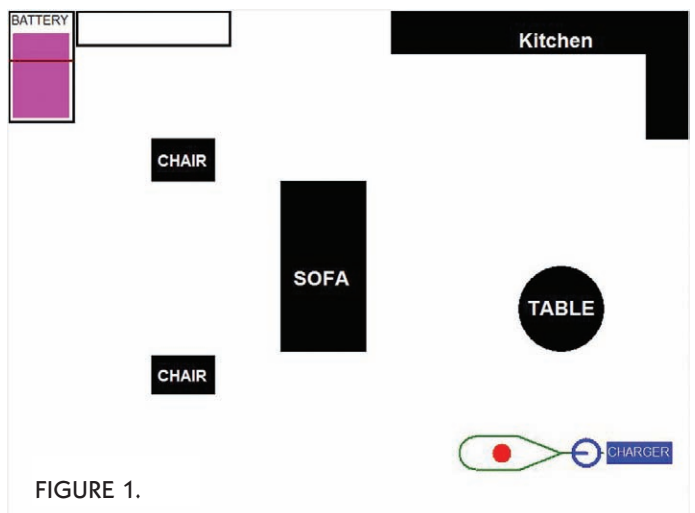
There are many aspects of a vacuum application that have to be solved before one can actually be built. First, some form of sensor is needed that can detect debris on the floor and pinpoint the room coordinates of its location. The first thing that comes to mind is a ceiling mounted web cam, but even fragments the size of popcorn could be hard to detect with a standard camera. Finding an appropriate sensor might require thinking out of the box. Perhaps some types of litter could be more easily detected with an IR or UV camera and it is not inconceivable that someone might come up with a totally new approach.

Even if a suitable sensor can be found or devised, there are other problems that must be considered. Once the robot knows where the dirt is, it must still navigate through the room to the desired destination while avoiding objects encountered along the way. The robot must also be able to determine when its battery is low and find its way to a charging station. Of course, people in the room should be avoided and changes in the placement of furniture should

not affect the robot's performance.

Developing an appropriate sensor technology could be expensive, so it might be comforting to know that the navigation problems were solvable before committing resources to developing the sensor technology. Corporations use simulation software to deal with this type of situation all the time.

For this project, RobotBASIC (a free language available from www.RobotBASIC.com) was used to create the simulated environment shown in **Figure 1**. It consists of a living space with a sofa, two chairs, a table, and a kitchen




```

ScrToCb    // copies the screen image to the clipboard
// now look for dirt in a 20 by 20 grid in the clipboard image
BmpFindClr "",LightBlue,sn,,.1,.1,20
// sn holds the number of the grid with the most 'dirt'
if sn>=0 // sn will be -1 if no LightBlue pixels were found
    // convert the grid number to a row and column
    Row=sn/20
    Col=sn#20
    // convert the row and column to screen coordinates
    dirtX=20+Col*40
    dirtY=15+Row*30
    gosub GoToDirt // go to the soiled area
endif

```

area. A meter in the upper-left corner displays the status of the robot's battery. The robot's charging station is located in the lower right corner of the figure. It is assumed that the robot must back into the charger in order to make the required electrical connections.

Our vision of the robot's behavior was straightforward. Normally, the robot should roam randomly throughout the room, hoping to vacuum the entire floor through accidental encounters. Periodically, the robot should acquire a picture of the room through a ceiling mounted camera. Although RobotBASIC does have commands for acquiring pictures from a web cam, this action was simulated by capturing the screen with commands designed for that purpose.

Once an image of the environment is obtained, the robot should search it for 'dirt.' If some is found, the robot should navigate to that area of the room and clean that area with appropriate movements. While all this is happening, the robot should monitor its internal battery. If the charge becomes low, the robot should temporarily abandon its current behavior and make its way to the charger. These actions of the robot may sound simple, but the details of carrying them out can be complex, as described below:

- How does the robot analyze the image of the room to find the dirt?
- Once dirt is located, how does the robot determine how to move from its current location to the debris?
- How does the robot find the charger and orient itself to obtain the charge?
- When the robot is moving to the soiled spot or to the charger, how does it deal with objects blocking its path?

Let's examine each of the above situations.

Locating the Dirt

Since our simulation assumes a sensor exists to detect the litter, we will assume that images of the room will show the dirt in a particular color. RobotBASIC has a command that divides an image into a user-defined matrix and supplies information about the amount of any specified color found in each of the matrix segments. Using this command, a program can determine the x,y coordinates of

a dirty area of the room. The code fragment in **Figure 2** shows how easily this can be done in RobotBASIC.

Moving to the Dirt

The simulated robot in RobotBASIC has commands that indicate the robot's position in the room. A real robot would have to

FIGURE 2.

obtain this information from a GPS or LPS (Local Positioning System). Once the robot knows the coordinates of the dirt and its own location in the room, trigonometry can be used to determine where the dirt is in respect to the robot itself. With this information, the robot can use its electronic compass (also available on the simulated robot) to turn to face the dirt.

If the room did not contain obstacles, then the robot could reach the soiled spot by simply moving forward. We will discuss how to handle obstacles shortly.

Finding the Charger

The charger could be found using methods similar to those used to navigate to the dirt. For the sake of diversity though, let's use a different approach. First, we will assume that there is a line on the floor that the robot can follow to the charging terminals (see **Figure 1** again). The line will have a loop on its free end so that if the robot finds the line and follows it, it will always end up at the charger. Furthermore, because the robot is following the line, it will always be pointed directly at the charging terminals when the charger is encountered. The robot can then connect to the charger by rotating 180° and backing up.

Of course, for this to work, the robot must be able to find the line. For that reason, a beacon (perhaps a cluster of IR diodes pulsing at a detectable frequency) will be placed on the ceiling over the line. The robot can face the beacon and then move forward looking for the line. During this movement — as with movement towards dirt — the robot must deal with obstacles that block its path.

Circumnavigating Around Obstacles

If you were walking through a room toward some destination and were blocked by an obstacle, what would you do? The most likely solution is to go around the object blocking your path and this is exactly what we want our vacuuming robot to do.

Whenever the robot encounters something in its way, it should randomly choose to turn left or right and use its perimeter proximity sensors to move forward while staying close to the object for some random amount of time. At

that point, the robot simply repeats its actions: face the desired destination; move forward until blocked by an object (or until the destination is reached); and (if an object is encountered) move around the object a random distance. If this sequence of actions is repeated over and over, eventually the robot will arrive at its intended destination.

This obstacle-avoidance behavior will be used when the robot is moving towards dirt or towards the beacon to find the charger.

Putting It All Together

Since RobotBASIC's simulated robot has all the sensors discussed in this article, it was easy to implement a program to test the algorithm described here. The robot behaves as expected, wandering randomly around the room unless the ceiling camera spots dirt. When the robot's battery runs low, it abandons its current behavior and makes its way to the charger, follows the line, and backs itself into position. When the charge is complete (as shown on the simulated meter), the robot continues the task that was interrupted by the need to recharge.

The program allows the user to use the mouse to sprinkle dirt around the room. There is even an option for letting the robot leave a trail to show where it has vacuumed. **Figure 3** shows how much of the room was actually covered after the robot spent some time cleaning.

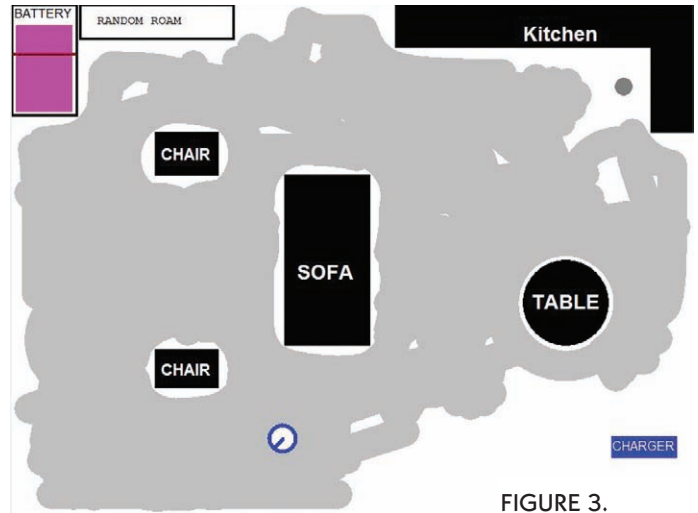


FIGURE 3.

Of course, you may have a totally different idea of how a vacuuming robot should operate. That is the beauty of simulations. You can improve on ideas or develop a totally new approach of your own. And, since RobotBASIC also has commands for controlling real robots, you will have a head start toward building a real vacuuming system.

The fully commented source code for the simulation described here can be downloaded from the APPLICATIONS page at www.RobotBASIC.com. **SV**

THE ORIGINAL SINCE 1994

PCB-POOL®

Beta LAYOUT

Servicing your complete PCB prototype needs :

- **Low Cost - High Quality**
PCB Prototypes
- **Easy online Ordering**
- **Full DRC included**
- **Lead-times from 24 hrs**
- **Optional Chemical Tin finish**
no extra cost

Watch "ur" PCB®
Follow the production of your PCB in **REALTIME**

FREE LASER STENCIL WITH ALL PROTOTYPE PCB ORDERS

Beta LAYOUT

email : sales@pcb-pool.com
Toll Free USA : 1 877 390 8541
www.pcb-pool.com

2006 TARGETS: **PCB-POOL** **PROTEUS** **RS-274-X** **Easy-PC** **Print Master**

CROSS the ROAD  **Cross the Road Electronics, LLC**
Cross-link Robot Control System

Finally, a control system for beginners, the developer and everyone in between.

Kit Includes:

- CANipede Robot Control Module (RCM)**
- 2CAN Ethernet Gateway**
- TRENDnet N Pocket Router**
- Cross-link/CAN Cables**

Cross the Road Electronics, LLC
www.crosstheroadelectronics.com

The NXT Big Thing #7

Ultrasonic

By Greg Intermaggio



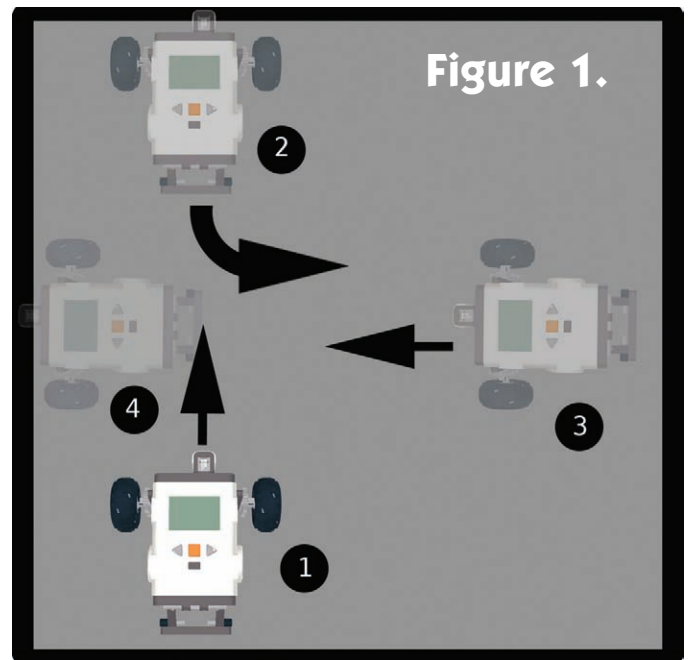
Last month, we built Eddie 2.0 and explored his new and improved design. This time, we're tackling the greatest, most competitive challenge ever devised for LEGO robots: Sumo!

Back in October '10, we covered a basic Sumo program — See The Light! Here's a quick review of what we learned.

You may remember **Figure 1** from that article. It depicts the path of a robot running on a standard Sumo program. The steps are:

1. Move forward until edge of ring.
2. Reverse for time, turn for time.
- 3 and 4. Repeat.

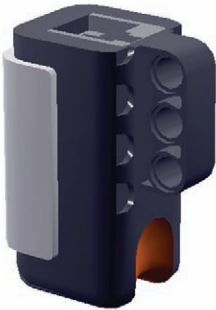
Simple enough, right? Today, we'll be revisiting Sumo, and adding a new feature to our program: ultrasonic opponent detection. I like to call it Ultrasumo. Let's start with the new build!



Building Ultrasumo

Plug the light sensor into port 3 and the ultrasonic sensor into port 2, and you're ready to go!

1. Start with a light sensor.



2. Add a double length friction pin, a 3x5 studless beam, and a 3x3 double peg as indicated.



3. Attach a 3x5 studless beam to the other side.



4. Add a standard length friction pin.



5. Snap on a 3x5 studless beam. (Note: the whole assembly will be a bit "floppy." Do your best to hold it together as indicated.)



6. Re-orient the assembly.

7. Attach a standard friction pin.



8.



Snap on a three-hole studless beam.

9.

Using just one double length friction pin, attach your ultrasonic sensor as indicated.



10. Add a three-hole studless beam to close off the other side.



11. Attach two standard friction pins.

Snap in a 3x5 studless beam to match the other side.

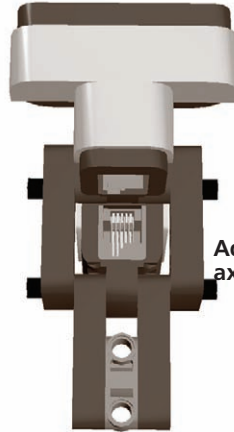


12.



13.

Re-orient the robot again.



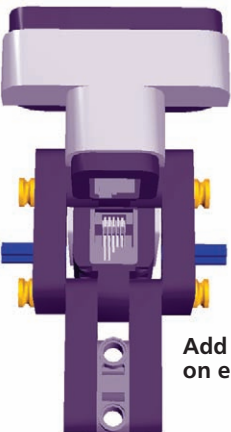
14.

Add two six-stud axles, as indicated.



15.

Secure them with half-bushings.

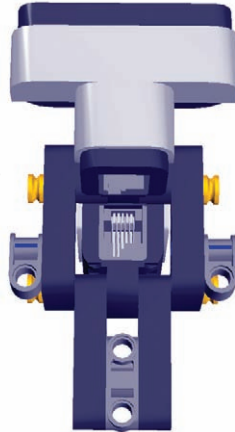


16.

Add a blue axle-pin on either side.

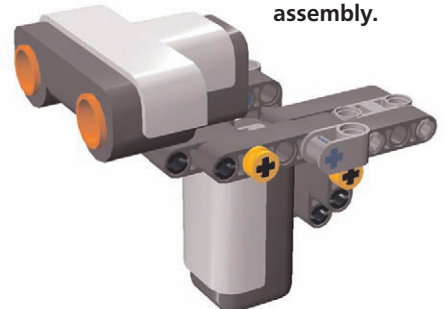
17.

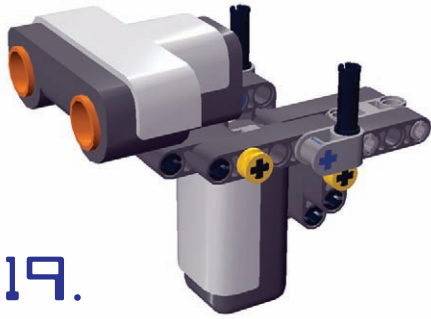
Slide in perpendicular axle-joiners as indicated.



18.

Re-orient the assembly.





19.

Snap in double friction pins.



20.

Attach three-hole studless beams.



21.

Connect standard friction pins to the studless beams.



22.

Snap the friction pins in place to Eddie's front as indicated (it may take a bit of creative jiggling).



23.

Re-orient the robot.



24.

Slide in axle extender friction pins to complete the attachment.

Introducing the Programming Challenge

Let's think for a minute about exactly how we'll be using the ultrasonic sensor. Eddie has two fundamental parts of his program:

- What to do normally.
- What to do when he reaches the edge of the arena.

When Eddie isn't seeing the edge of the arena, he's simply moving forward so that if he hits the opponent robot, it will be pushed to the edge (and outside) of the Sumo ring. Because he's already moving forward at full power, there's no reason to use the ultrasonic sensor as he does so, which brings us to the other part of his program.

When Eddie is backing up and turning, he could very

well be passing the opponent robot and/or missing an opportunity to charge! It's during this phase of the program that we'll make use of the ultrasonic sensor.

Tackling the Program

This will be a relatively complex program compared to those we've worked on in the past. There are many ways to go about solving a programming problem and, as usual, I encourage you to explore your own ideas! That said, here's what I did.

Two quick notes:

- My durations are based on Eddie 2.0 with a 1:5 (8:40) gear ratio.
- I used a dark arena floor with a light ring. If your floor is light and your ring is dark, you'll want to switch the > sign in the light sensor switch to a < sign.

Eddie Should Now Be Ready To Rock!

Download your program and give it a test. He should go forward full speed until he hits the edge of your arena, then

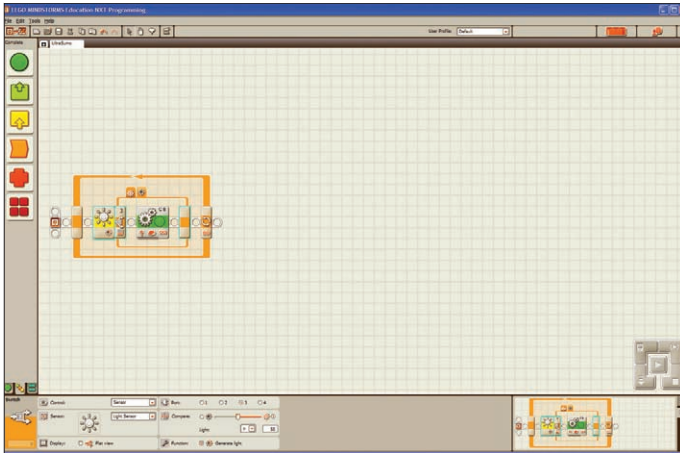


Figure 1. Start with an infinite outer loop and a light sensor switch (uncheck "Flat View") from the flow palette, with a "Move" command in the dark tab of the switch. Set the port to 3, and adjust the values based on your sensor readings.

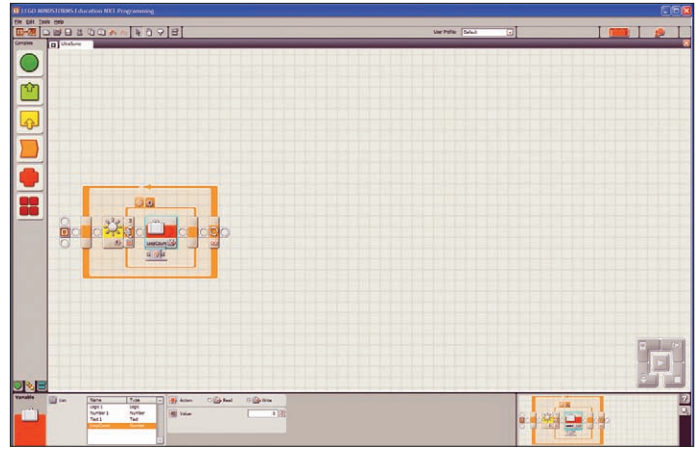


Figure 4. Select "LoopCount" and "Write."

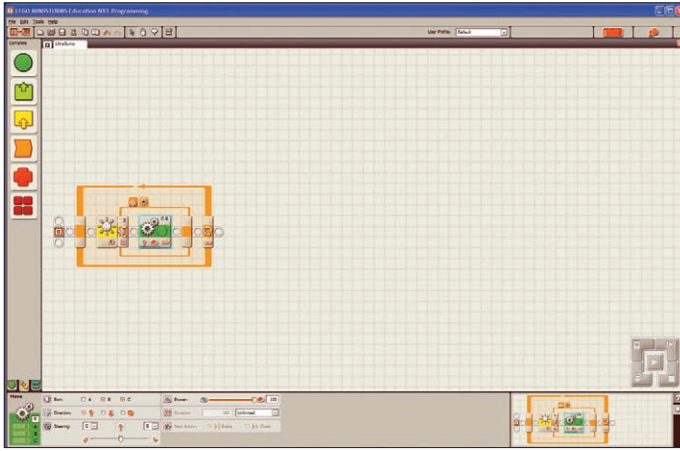


Figure 2. Set motors C and B forward for an unlimited duration.

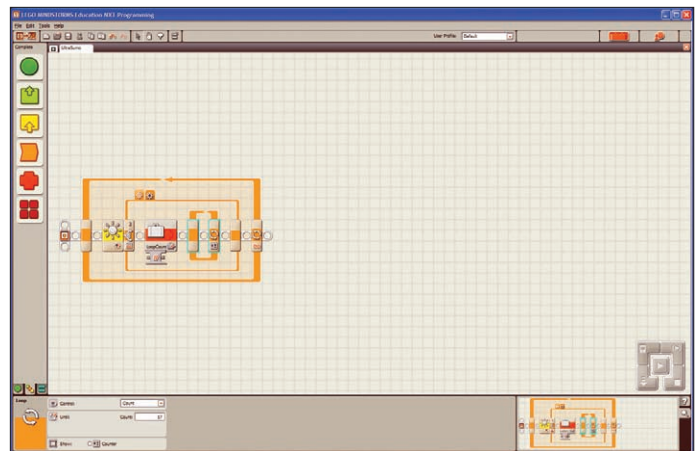


Figure 5. Add a loop and set Control to Count. Count to 17. This means that it will loop 17 times. We're going to break up the back-up and turn into 17 smaller movements, and check the ultrasonic sensor after each one. That way, if Eddie sees the opponent robot mid-movement, he can react immediately.

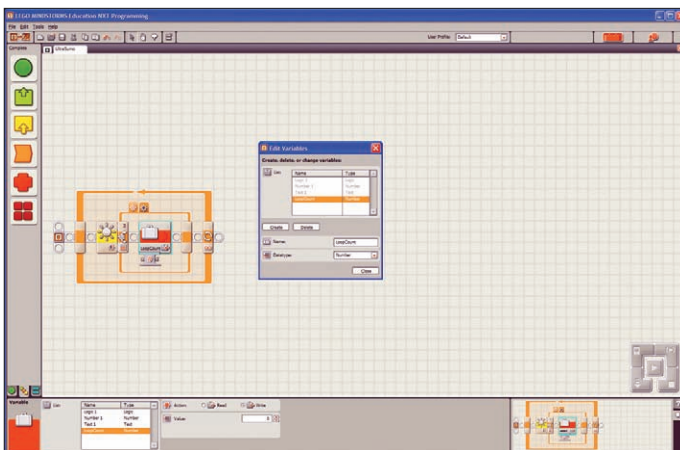


Figure 3. Click to the light side of the switch. Go to Edit > Define Variables and create a number variable called LoopCount. Then, from the data palette, drag in a variable block.

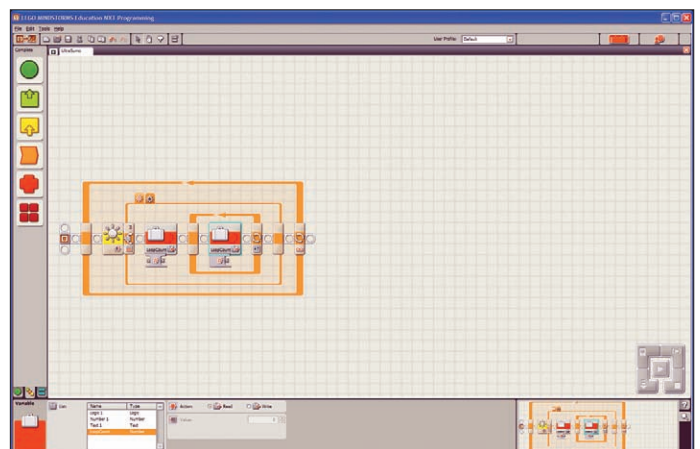


Figure 6. Add a variable block inside the new loop; select LoopCount and Read.

reverse and turn. You'll notice that as he reverses and turns, he'll move in chunks. After each of these short movements, he's checking his ultrasonic sensor to see if his opponent is in front of him. If all goes well, he should immediately charge another robot, your hand, or even a cat that walks in front of the ultrasonic sensor.

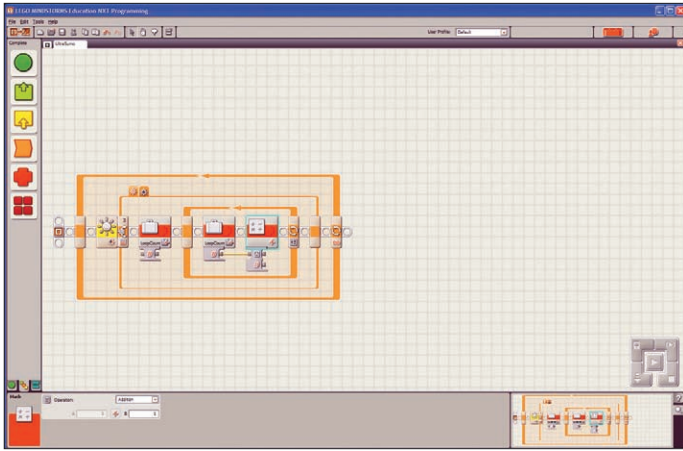


Figure 7. Add a math block from the data palette and run a data wire from Value on LoopCount to A on the math block. Select the math block. For Operation select Addition, and for B, type in 1.

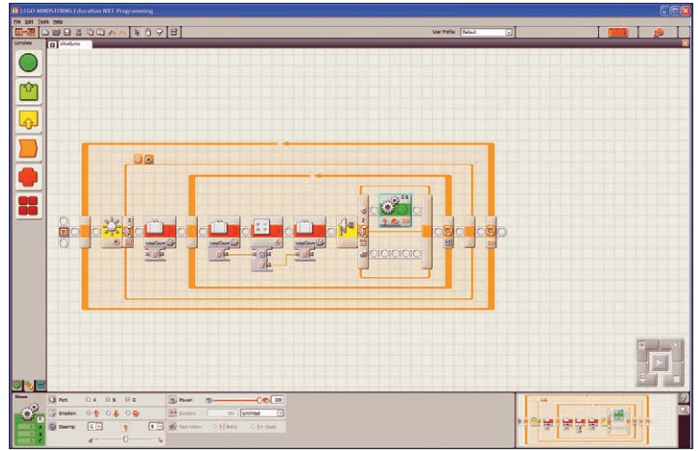


Figure 10. For the < eight inches side of the switch, add a motor block that will go forward at full power for an unlimited duration. This is what we want Eddie to do when he sees a nearby robot.

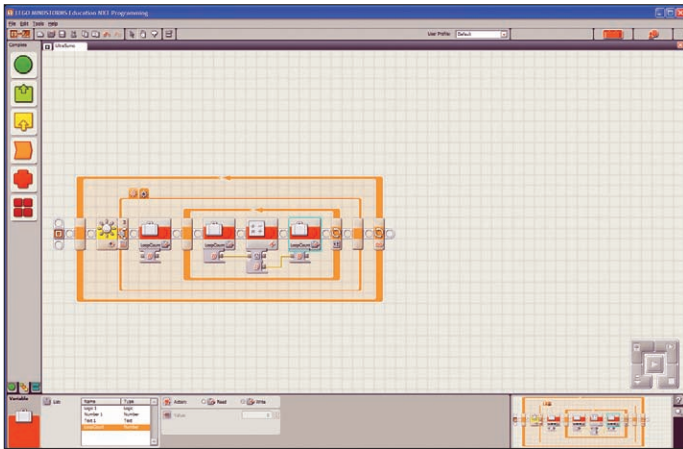


Figure 8. Add a variable block and select LoopCount again. Choose Write, and run a data wire from the result of the math block to the value of the variable block. This will add 1 to our LoopCount variable each time our loop runs, giving us a way to track it.

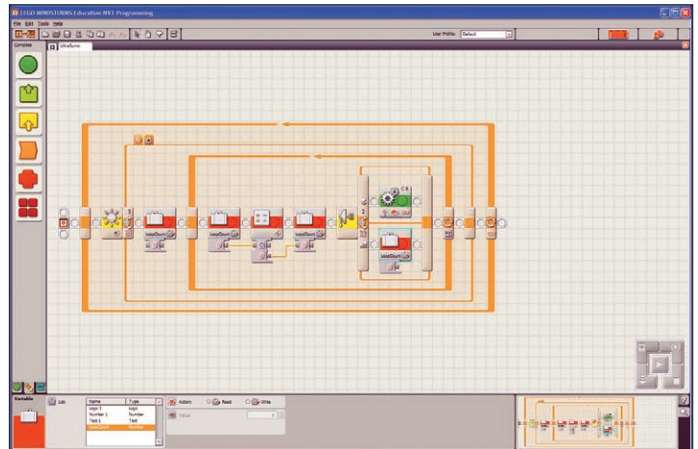


Figure 11. On the > eight inches side of the switch, add a LoopCount variable.

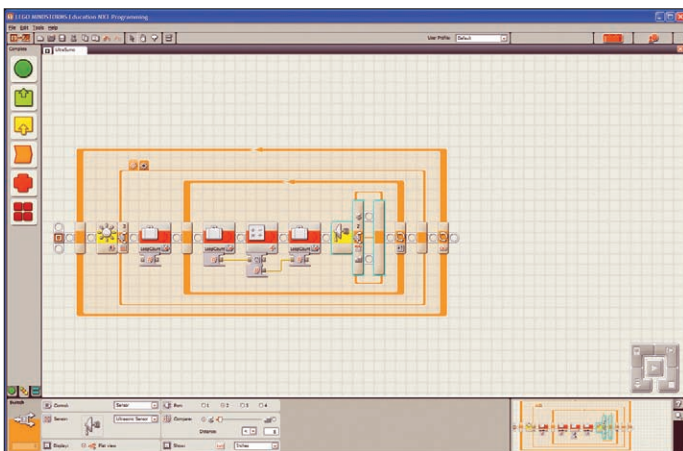


Figure 9. Add a switch and select Ultrasonic Sensor. For port, select 2. For distance, select < eight inches.

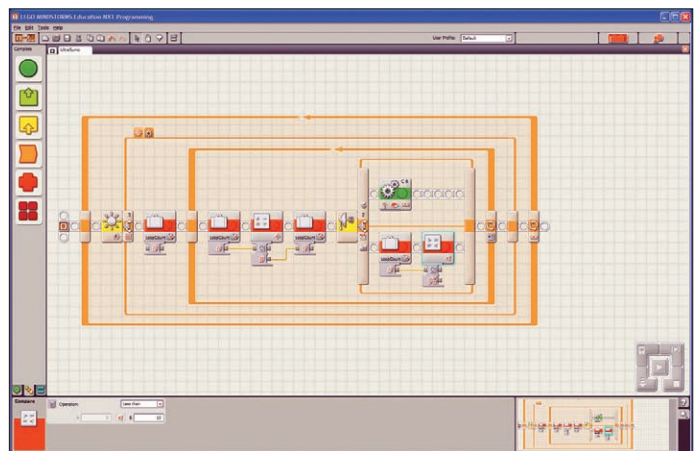


Figure 12. Add a compare block from the data palette. Select <. Connect a data wire from Value to A, and for B, type in 10.

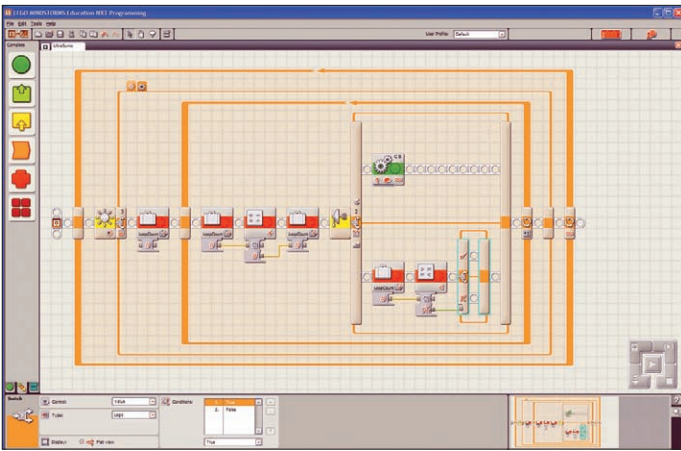


Figure 13. Add a switch and for Control, select Value. For Type, select Logic. Attach a data wire from the result of the compare block to the input of the switch.

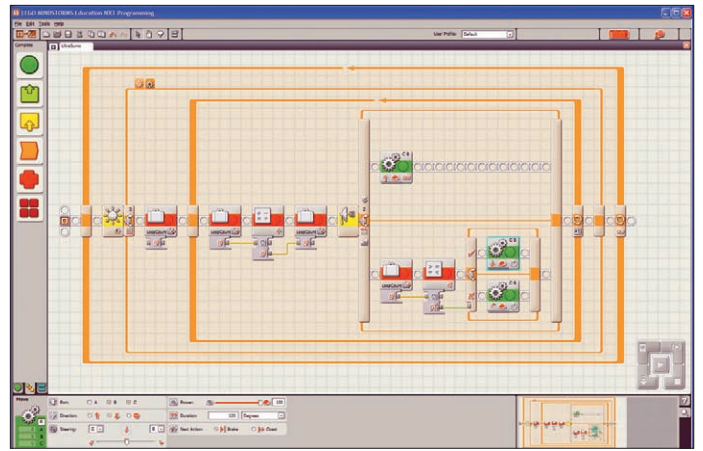


Figure 15. On the True side of the new switch, add a move block — full power backwards on B and C for 120 degrees. This is what Eddie will do for loops 1-10.

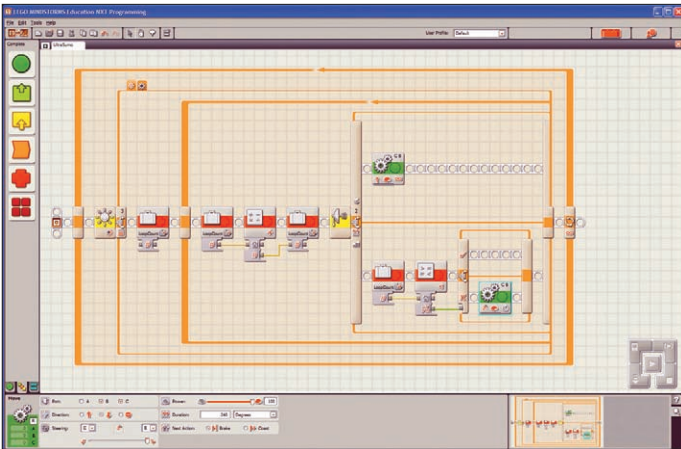


Figure 14. On the False side of the new switch, add a move block. Set B and C backwards full power in a full right turn for 240 degrees. This is what Eddie will do for loops 10-17.

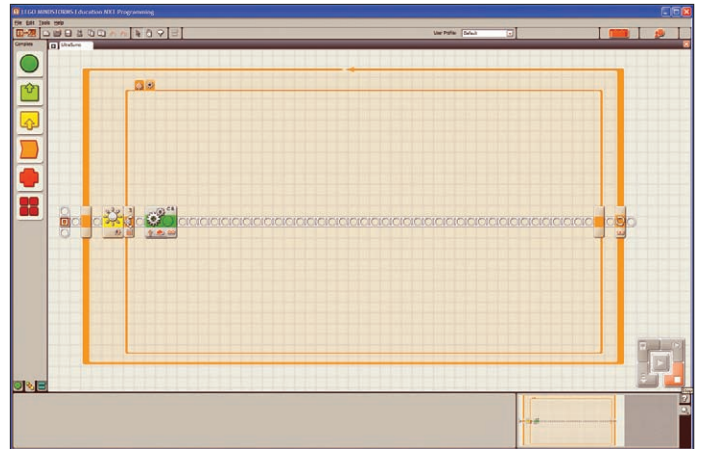


Figure 16. Finally, on the dark side of our light sensor switch, add a move block that sends Eddie forward at full power.

More Challenges!

Now that you've seen how to program Eddie for Ultrасumo, here are a few ideas to take things further!

- Annoyed with having to change your light values every time you show Eddie off at a new venue? Try programming him to self-calibrate! You'll have to record a value for dark and a value for light using variables, and find the average to use as your threshold.
- Try rewriting the Ultrасumo program in a new way that will have the same output.
- Create an attachment to hold 2 ultrasonic sensors and program Eddie to follow his opponent!

Wrapping Up

This month, we revisited Sumo and used an ultrasonic sensor to improve our program. We used an array of loops, switches, and mathematical operations to get our program working.

Next month, we'll be exploring the compass sensor, available from HiTechnic. Pick up a sensor or three from their store at www.hitechnic.com and stay tuned for the next installment of our NXT Big Thing! **SV**

www.servomagazine.com/index.php?/magazine/article/february2011_Intermaggio

Build Your Own Big Walker Part 1

by Daniel Albert

www.servomagazine.com/index.php?/magazine/article/february2011_Albert

Ever since I started working with bipeds, I have dreamed of building a full size humanoid robot.

Unfortunately, there just aren't any kits around. I believe this is because it's just not that easy to build a stable biped. Even small bipeds suffer problems of stability, processing control, power consumption, and mechanical slop. As we scale up the size, we scale up the problems and the cost.

Today's typical small bipeds (height range of about 8" to 18") cost approximately \$400-\$2,000. Custom-built models for larger sizes even under 36 inches can easily hit \$10,000. Human size bipeds typically cost many hundreds of thousands to many millions of dollars.

I set out to show that it is possible to build a full size biped that can slowly walk for well under \$10,000.

Is there a low cost, simple solution to build a full size biped robot without a huge capital investment and a group of rocket scientists?

I would like to remind all of those who would say "No, there isn't" that in the 18th century, John Harrison — a self-educated maker of wooden clocks — built the first very accurate ship's chronometer and won a 10,000 pound (that's old British money) prize. Highly educated scholars of the time refused to acknowledge him because he didn't

have a formal education. So, don't despair if you are not highly trained in the "rocket science" disciplines. Success follows determination and the "Never give up, never surrender" attitude.

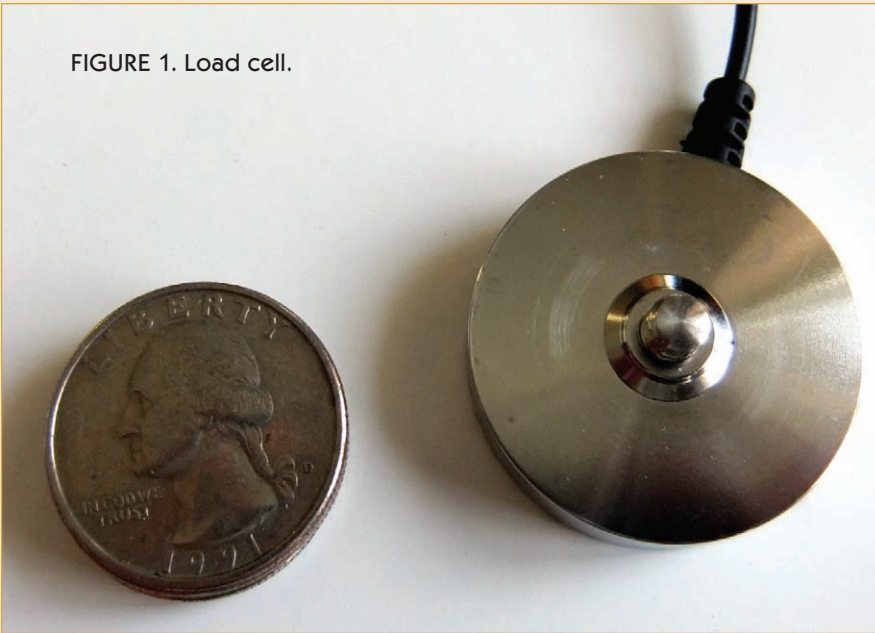
By the way, I have met some very smart, highly educated people who see some problems as so unsolvable that they never even try. Well, I'll try.

Actually, I have been building this bot for several years now in my spare time away from my day job. I built some prototypes of the foot to prove my theories of biped control. These models were so successful I decided to start building the complete robot. It will take a while to properly explain each component, so I have broken them down into four segments:

1. Theory of operation. Balancing with load cells, lightweight feet, limb processing board.
2. Mechanical design. Weight distribution, foot linkage, belt driven gear box, power system.
3. Distributed processing. Multi-processor control system, distributed communications.
4. Walking. A synergistic fusion of static frames and dynamic adjustments.

I've made some changes to the traditional bipeds that are seen on the market today.

FIGURE 1. Load cell.



most difficult part is how to calculate the CG of the weight on each foot.

If I can do it, so can you.

Balancing With Load Cells

Every biped robot needs some sense of balance. When I started working on this bot, my day job was writing firmware for highly accurate load cells. I am a firmware consultant by trade and I was fortunate to find a client that created the exact sensor I needed for my biped.

Load cells are the devices used in scales to determine weight. To be useful in this project, I need accuracy and fast feedback. The load cells I used are about .2% accurate over full scale (0-10 lbs) or about 1/3 of an oz. They give a new reading once every millisecond. The cells

Gone:

1. Big flat square feet.
2. Heavy servos for foot and ankle movements.
3. One MCU controlling the whole robot.

Added:

1. Human foot style with three points of contact.
2. Linkage from the upper leg to control foot position.
3. One controller for each limb; Limb Processing Board (LPB).

are mounted on the bottom of the foot as contact points in a triangular layout similar to the human foot.

The human foot has 11 points that distribute the weight. Three are extremely critical. The heel, the ball of the first metatarsal at the big toe, and the fifth metatarsal at the pinkie toe form a tripod. There may be better designs for robotic walking but with this one, I can relate to the forces and feedback the robot will experience.

The first prototype was a one leg, one foot design with three load cell sensors and three servos. Two servos controlled the two axis of the heel and one for the knee. The controller and battery pack were mounted atop the structure.

The goal is to have the foot always support the load within the CG. The CG is the average location of the weight of the object. When the bot is not moving and the CG is inside the triangle of the three sensors, it is stable and resists the temptation to fall over and destroy many hours of work. The LPB reads the weight of each sensor every millisecond and creates a running average for each point. Once every five milliseconds, it calculates the most stable point within the triangle as the target CG. The error between the current CG and the target CG is fed to a PI (Proportional Integral) loop. The PI loop self-adjusts the servos and the CG is maintained within the triangle.

This inverted pendulum technique worked well for very slow movements. I could move the base of the foot and the servos would keep the weight within the triangle.

Lightweight Feet

I could have built the complete biped from the prototype design. In retrospect, I would have achieved a slow shuffling biped sooner had I done so. The problem is in the slop and play of each joint and the flexing of the connecting linkage. Unlike biological muscles, servos have gears and bearings. Each one has a little play. The longer the limb, the more that play translates to big movements at

Theory of Operation

There are many different theories and models used to control bipeds. Some are extremely sophisticated and require some very complex mathematics to control the dynamic forces we take for granted when we walk. This article is NOT about that. There will be some control theory presented, but I will try to keep it to a minimum. The gist of the theory is that with good enough sensors, we can build a large walking robot based on the principle of "center of gravity." Even though I knew the first prototype would never be able to run, I wanted to create a design that could evolve into a running biped. To achieve this, some of the standard "cheats" commonly seen in smaller bipeds have been designed out.

It's hard to fall over if you're wearing cement overshoes. A lower center of gravity (CG) will give higher stability. However, humans have a high center of mass. When we walk, we throw our weight outside our CG and catch ourselves with the other foot. Then we bound out again on the next step. For these fast movements, an inverted pendulum where the mass is above the pivot point is employed.

This is probably the best way to build a fast moving biped, but it does take some "rocket science."

What I am proposing here is a simple way to build a large walking biped without the heavy mathematics. The

the opposite end. The more weight at the end of that limb times the length creates uncontrollable oscillations in the system. Two things help. Tighten up the slop and lighten the load.

The tolerances of the servos are fixed. I was doing my best as a hobbyist to build tight linkage but in the end, the most bang for my buck was the weight at the end of the limb. So, I decided that the servos at the foot had to go.

I mounted the two foot servos at the knee. One controls the first metatarsal of the foot, the other controls the fifth metatarsal of the foot. The heel joint just rotates. The linkage pulls and pushes the other two points allowing for foot leveling. Small movements are key as it does not take much movement in the servo to change the CG. The servos are fast and the LPB has hardly anything to do except work at keeping the weight inside the triangle. I will go into more details of this in next month's segment.

Limb Processing Board

When I started this project a few years back, there were not as many controller boards as there are today. I could not find one that had the interfaces I needed to read the sensors, control the servos, and work in a fast distributed system. I had played with ExpressPCB a few times before and found it very easy to build a board with everything just the way I wanted it. I designed a board around the Microchip dsPIC30F4012. It has plenty of processing speed, comes with a free compiler and lots of libraries to do PI calculations in real time. Microchip has a plethora of new devices every year and at some point I will probably redesign the board with something faster than the I²C bus I use to talk between boards. For now, it does the job.

I use the same program for each LPB. One is designated as the master and the others as slaves. There is a command line interface that can communicate wirelessly with a PC's terminal window via a Digi XBee board. These boards have a simple-to-use serial interface. I only talk to the master.

Next Month

In the next installment, I will move away from the prototype to show the mechanical design of the full size biped. I will detail the weight distribution, foot linkage, belt driven gear box, and the power system. **SV**

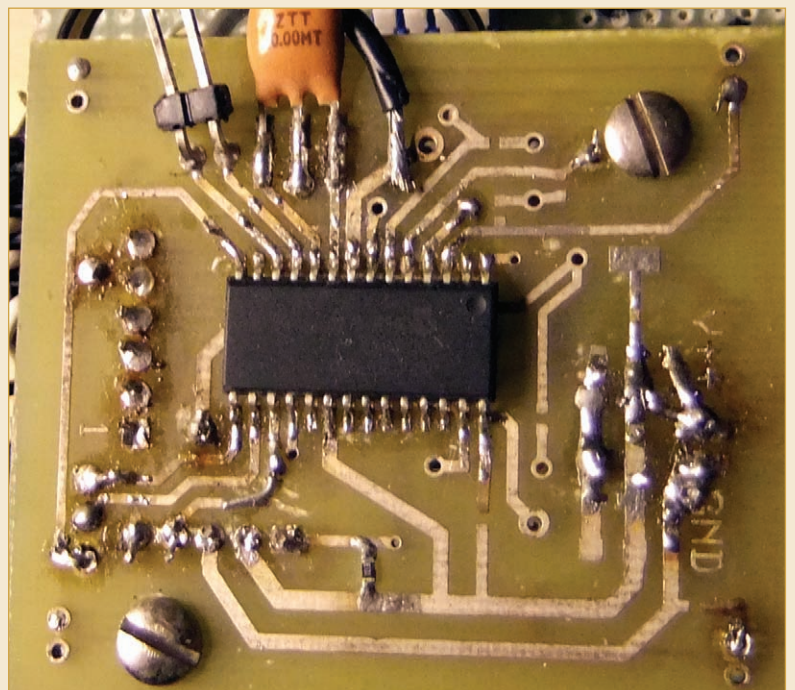


FIGURE 2. My foot prototype.



FIGURE 3. Standing position.

FIGURE 4. The LPB.



Control Your Steel Army With the RPM3

by Fred Eady

RF is magic and I believe that all RF engineers graduated from the University of 666. Despite my prejudice, I still love doing embedded RF projects. The embedded radio we're going to talk about this month can be considered the FEDEX of UHF radios. If a packet absolutely has to get there, one of the RPM3 series of UHF radio packet modems is driving the big white truck.

The RPM3 (Radio Packet Modem 3) in our possession is technically identified as an RPM3-914-17 UHF radio packet modem. We'll call it RPM3 for short. As you will see as we move along, the RPM3 is not your average embedded radio. However, we can use all of the embedded microcontroller platforms we've built over time to host an RPM3. Before we look at what we will stand behind an RPM3, let's examine the RPM3 in detail.

Have Packet, Will Travel

The RPM3 is an intelligent radio packet modem that can communicate in point-to-point, point-to-multipoint and broadcast modes. The RPM3's brains are obvious to the most casual observer in **Photo 1**. I'd say that the microcontrollers we captured thinking in **Photo 1** are responsible for the RPM3's ability to address a data packet and guarantee its delivery. Guaranteed packet delivery entails the process of checking for errors and acknowledging the reception of a good packet. If the packet is damaged in transit, the RPM3s that are involved directly in the communications link can retransmit the crippled packet. The RF business end of our RPM3 is captured in **Photo 2**.

Basically, if the embedded host can tap out and recognize signals that look like UART RS-232 data, that host can work with an RPM3 to form one portion of a wireless serial link. Any number of RPM3-equipped devices can be interconnected. All of the low level packet assembly, error checking, and RF control are done for us thanks to the pair of PIC microcontrollers we photographed in **Photo 1**. The larger PIC16LF876A's maximum clock speed is 10 MHz with a supply voltage

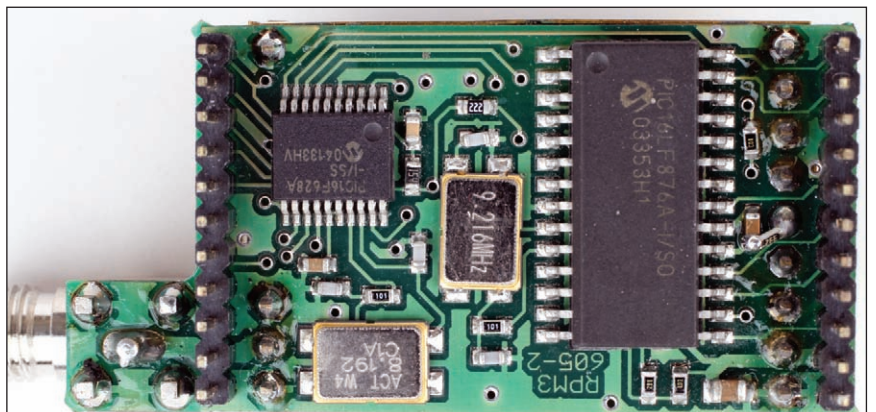


PHOTO 1. Not one, but two PIC microcontrollers are in control of the magic RF stuff under the hood in **Photo 2**.

that is greater than +3.0 volts. The PIC16LF876A is the execution unit that supports the RPM3's radio packet modem subsystem which is clocked by a 9.216 MHz crystal. The radio packet modem directly interfaces with the host microcontroller. The smaller PIC16F628A has the same 10 MHz clock frequency characteristics when operated at +3.0 volts. However, the PIC16F628A can operate with clocks up to 20 MHz when powered with a +5.0 volt power source. Theoretically, the PICs in our five volt RPM3 could operate at +3.0 volts. The caveat is that we really don't know if the RF section is voltage sensitive. The PIC16F628A clocks mnemonics that form the Fast Radio Packet Controller which interfaces directly to the radio packet modem on one side and the actual RF transceiver on the other. The controller is serviced by an 8.192 MHz crystal.

Figure 1 sums up the RPM3 subsystems. The bottom line is that whatever data we pump into the transmitting +5 volt powered RPM3 has a very good chance of being delivered to any other desired RPM3 that is in receiving range.

RPM3 Modes and Pins

Figure 2 provides us with the lay of the land as far as the RPM3 is concerned. The active-low WAKE/DTR pin will force the RPM3 into shutdown state when it is driven logically high. The RPM3 is not accessible while in shutdown state. We will assign a host microcontroller I/O pin to the WAKE/DTR input as the shutdown state is actually low power sleep mode.

The RPM3 will automatically enter the standby state when powered up. Standby state is maintained until a connection request is received. A host can initiate a connection request by simply transferring data to the RPM3 via the serial interface. Thus, a logical connection is established between the host and

FIGURE 2. This is vital information. This graphic also sheds some light on the RPM3 block diagram in **Figure 1**.



PHOTO 2. As you can see, our RPM3 is a five volt version that is tuned to the North American 900 MHz frequency.

end device. Once the logical connection is established, the contacted RPM3 enters a connected state which allows data to be transferred.

Serial data is transferred between the embedded host and RPM3 via the RPM3's TXD and RXD pins. Data flow control can be implemented by monitoring the RPM3's CTS pin. The RPM3 will take its CTS line logically high when its serial receive buffer reaches 66% of its capacity. When the serial receive buffer empties to 33% of capacity, the RPM3 will drop the CTS to a logically low level. Our host microcontroller has enough extra I/O pins available to allow us to dedicate one to the RPM3's CTS output.

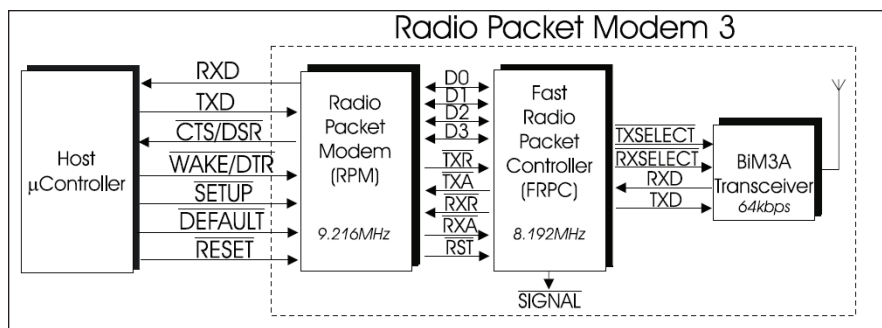
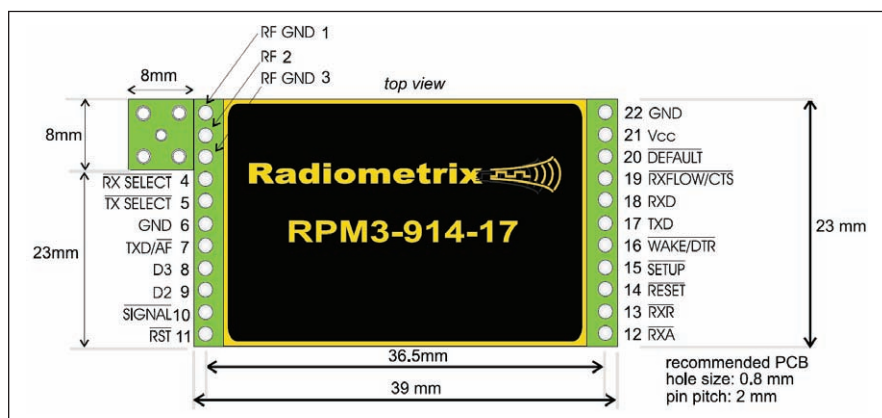


FIGURE 1. This graphic depiction of the RPM3 pretty much says it all. The BiM3A transceiver easily handles the RF traffic as the RPM3's maximum throughput is 28 kbps.



The lower case “r” that precedes each definition is insurance just in case the pseudo name is a reserved word that the C compiler may choke on.

Naturally, we’ll use the MAL (Microchip Application Libraries) to help us establish a USB link between our laptop and the RPM3 by way of the PIC18F4550. Thus, the PORTB definitions we cranked out will reside in the *HardwareProfile.h* file. While we’re at it, let’s go ahead and write the PORTB initialization code and add it to the LED init code:

```
#define mInitAllLEDs()    LATB = 0xFB; TRISB
= 0xC8; LATE = 0xFC; TRISE = 0xFC;
```

The *mInitAllLEDs* macro will drive all of the RPM3’s active-low lines logically high at power up or reset. The RPM3’s CTS is an input for the PIC18F4550 and that explains the TRISB value of 0xC8 instead of 0xC0.

Since the RPM3 is designed to interface in a true RS-232 fashion, we’ll need to logically lay out the PIC18F4550’s USART subsystem. We’ll include the DTR and RTS modem control lines just in case we need to deploy them:

```
#define UART_TRISTx      TRISCbits.TRISC6
#define UART_TRISRx     TRISCbits.TRISC7
#define UART_Tx         PORTCbits.RC6
#define UART_Rx         PORTCbits.RC7
#define UART_TRISRTS    TRISBbits.TRISB5
#define UART_RTS        PORTBbits.RB5
#define UART_TRISDTR    TRISBbits.TRISB2
#define UART_DTR        PORTBbits.RB2
#define UART_ENABLE     RCSTAbits.SPEN
```

Note that the PIC18F4550’s DTR modem control I/O line is also the RPM3’s WAKE/DTR input. We’re ready to put some code together. Let’s begin by defining some variables we’ll need:

```
unsigned char counter;
unsigned int  rloopcounter,rloopcounter1;
```

Naturally, we’ll need to initialize these counters before we can go loop-de-loop with them:

```
counter = 0;
rloopcounter = 0x8FF;
rloopcounter1 = 0xFF;
```

The variable initialization takes place in the *User Init* function which resides inside of our modified main source file *rpm3.c*. We’ll also need the services of the PIC18F4550’s USART. The CDC demo we’re basing our RPM code

PHOTO 3. Here’s my version of an RPM3 radio node. I did some pin stretching to mate the RPM3’s 2.0 mm pitch pins to the 0.1 inch pitch breadboard. I used wirewrap and point-to-point soldering techniques to make the physical connections.

on sets up to run the USART at 19,200 bps. The RPM defaults to 9600 bps. So, we’ll have to do some calculations and load the PIC18F4550’s USART to run at 9600 bps. I’ll let you reference the PIC18F4550 datasheet for that baud rate formula. When you find it and do the math, you should end up with an SPBRG value of 311 or 0x137 hexadecimal. Here’s the USART baud rate code behind the math:

```
UART_TRISRx=1;    // RX
UART_TRISTx=0;    // TX
TXSTA = 0x24;     // TX enable BRGH=1
RCSTA = 0x90;     // Single Character RX
SPBRG = 0x37;     //0x0137 for 48MHz ->
                  //9600 baud
SPBRGH = 0x01;    // 0x0271 for 48MHz ->
                  //19200 baud
BAUDCON = 0x08;   // BRG16 = 1
c = RCREG;        // read
```

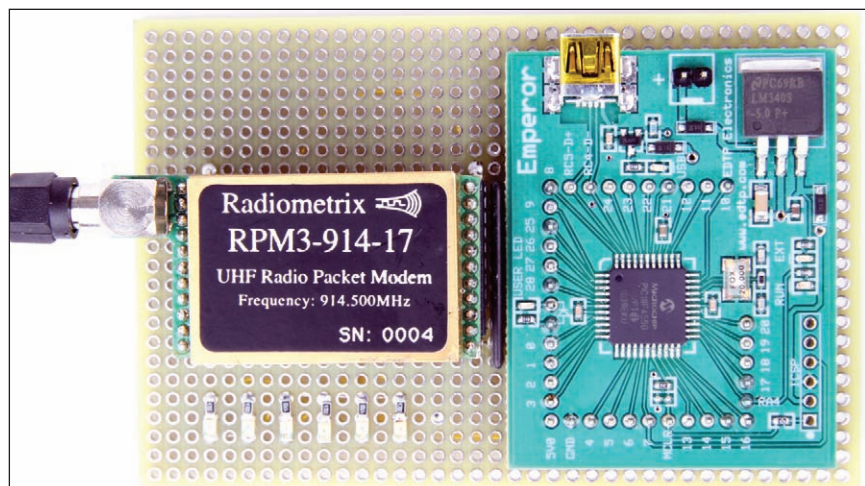
Okay. As you can see in **Photo 3**, our hardware is ready to go. It’s time to perform the mandatory magic smoke test. If the magic smoke remains in its designated quarters, we’ll get to work writing and testing some RPM3 functions.

Three Days Later

No magic smoke in the air and nothing is working. Hmmmm ...

Just a few minutes ago, I was scratching my head wondering what in the world was wrong. The schematic matches the electronic construction and the Emperor has been proven to work in a previous discussion. This is kinda like pulling out the oscilloscope because you’ve tried everything else. So, before I hit the PANIC BUTTON, I pulled out the RPM3 datasheet and started to absorb each and every word in the hope that I had missed something simple.

AHA! Take a good close look at **Figure 3**. The RPM3 is logically a DCE device. That means the RPM3’s transmit pin is actually an input. That makes its receive pin an output. If the term DCE and the logical swapping of the RPM3’s transmit and receive lines makes no sense, think of the RPM as a modem. The data that the modem “receives” is transmitted to the



Pin name	Pin	Pin Function	In/Out	Description
RF GND	1, 3	RF signal ground		BNC casing/coax braid connection
RF	2	RF signal	Input or Output	Antenna pin/coax core connection
RXSELECT	4	Receiver Select	Input or Output	Internal RF Receiver Enable to BiM3A or RF Receiver Active Indicator
TX SELECT	5	Transmitter Select	Input or Output	Internal RF Transmitter Enable to BiM3A or RF Transmitter Active Indicator
TXD/AF	7	Transmitted Data or demodulated signal	Input or Output	Transmitted Packetised Data to BiM3A Analogue Demodulated signal from BiM3A
D3	8	FRPC Data line	NC	Internal data line between RPM and FRPC
D2	9	FRPC Data line	NC	Internal data line between RPM and FRPC
SIGNAL	10	Preamble Detect	Output	Valid preamble indicator
RST	11	FRPC reset	NC	Resets FRPC which also isolates BiM3A
RXA	12	Receive Acknowledge	NC	RPM to FRPC download Request Acknowledge
RXR	13	Receive Request	Output	Valid Data packet indicator
RESET	14	Reset	Input	Hardware reset of the RPM3
SETUP	15	Enter Setup	Input	Enter RPM3 configurator after a RESET
WAKE/DTR	16	Wake or Shutdown	Input	Wakes RPM3 when low, shuts down when high
TXD	17	Serial transmitted data	Input	Host (DTE) to RPM3 serial transmit data
RXD	18	Serial Received data	Output	RPM3 to host (DTE) serial received data
CTS	19	Clear To Send	Output	Hardware flow control of data from host (DTE)
DEFAULT	20	Force 9600bps	Input	Force the RPM3 serial interface to 9600bps
VCC	21	Vcc Supply	Input	+5VDC or +3VDC regulated supply
GND	6, 22	Ground	-	Supply Ground internally connected to GND

FIGURE 3. This falls into Fred Eady's Second Rule of Embedded Computing, which says "Never assume anything."

```
if(rloopcounter == 0)
{
    rloopcounter = 0x08FF;
    -rloopcounter1;
}
-rloopcounter;
```

The variable *rloopcounter* is preloaded with 0x08FF and *rloopcounter1* is loaded with 0xFF before the loop is initially executed. The idea is to give us enough time to get the terminal emulator up

host or DTE device. On the other hand, what the modem "transmits" is received from the DTE device. And to add insult to injury, duh huh ... the "M" in RPM3 stands for MODEM. With that minor detail out of the way, let's take a look at how I got the RPM3 to spit out the Start Message you see in **Screenshot 1**.

Since we're running the PIC18F4550 as a serial emulator, the virtual COM port does not appear until the PIC18F4550 enumerates. So, we can't be standing there waiting for a message with the terminal emulator running. The trick is to get the PIC18F4550 enumerated, let the virtual COM port initialize, and wait. In that the CDC serial emulator code is constantly rotating through multiple tasks, all we have to do to generate a delay is do some adding and subtracting on every pass:

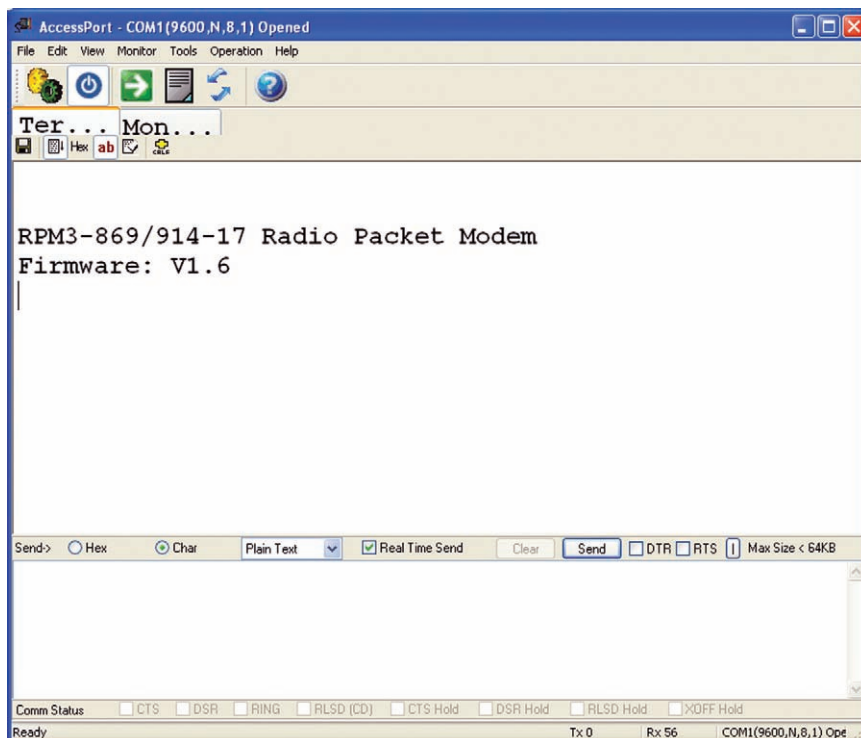
before we do anything to the radio.

The first thing we want to do is make sure we can tickle a Start Message out of the RPM3. That can easily be done by simply resetting the RPM3. Resetting it entails dragging the RPM3's RESET line logically low for a period of time and releasing it to a logical high:

```
if(rloopcounter1 == 0 && counter == 0)
{
    rRESET = 0;
    counter = 1024*1024;
    while(--counter);
    rRESET = 1;
    counter = 1024*1024;
    while(--counter);
    counter = 1;
    rloopcounter1 = 0xFF;
}
```

Once the *rloopcounter* loop falls through, we pull the RPM3's RESET line logically low and kill some time with the *counter* variable. Before we leave the RPM3 reset function, we make sure that the function's code will only run once per PIC18F4550 reset by setting the *counter* variable to 1. We also need to reload the *rloopcounter1* loop variable to make sure the kill-some-time loop continues to run as designed. A very short time after the RPM3's RESET pin is set to a logical high, the Start Message in **Screenshot 1** is transmitted to the PIC18F4550 USART which, in turn, directs the Start Message to our terminal emulator via its USB port.

At this point, we should be able to



SCREENSHOT 1. This is called the Start Message. The Start Message is generated on every power-up or reset. We can use the RPM3's built-in configurator to display or suppress the Start Message.

converse with the RPM3. Let's code up a mess of C to put it into configuration mode. To enter configuration mode, the RPM3's SETUP pin must be logically low following a reset. That's pretty obvious in this SETUP code snippet:

```
if(rloopcounter1 == 0 &&
counter == 1)
{
    rSETUP = 0;
    rRESET = 0;
    counter = 1024*1024;
    while(--counter);
    rRESET = 1;
    counter = 1024*1024;
    while(--counter);
    counter = 2;
    rloopcounter1 = 0xFF;
}
```

Screenshot 2 captures the results of our RPM3 reset and the entry into configuration mode. Note that the RPM3's serial number is displayed. The serial number is used to uniquely identify each RPM3. I happen to have serial numbers 4 and 5.

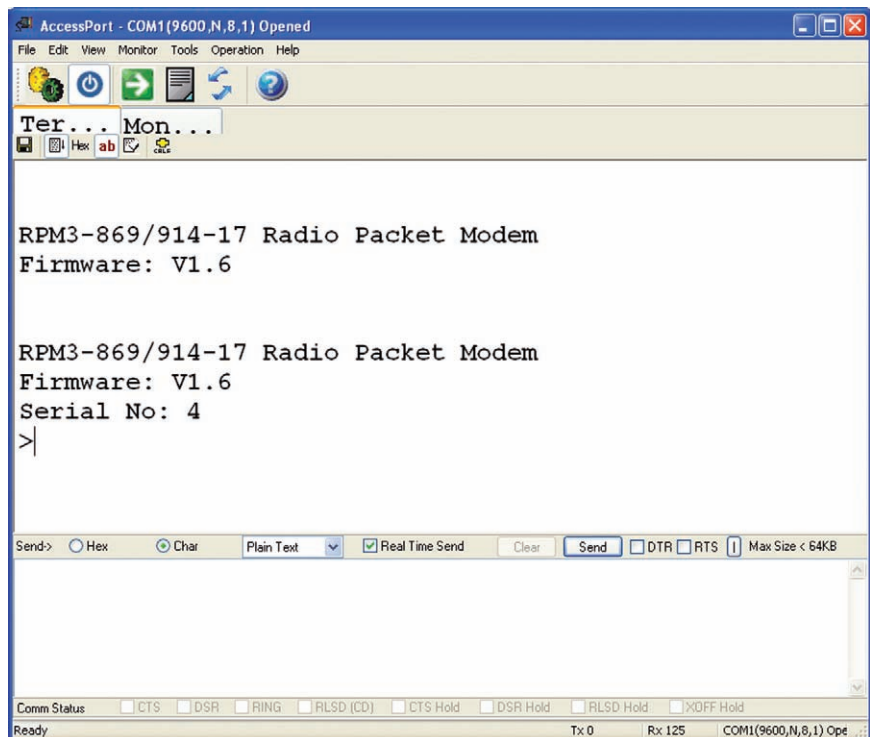
I manually entered the CONFIG command in **Screenshot 3**. The results are default settings. Maximum throughput may not be desired if the modems must communicate frequently. The maximum throughput setting works best when large amounts of data need to be moved quickly. The RPM3 transmits in slots otherwise. The *slots* and *slotsw* settings allow other modems to have a chance to seize the air for short message deliveries.

Individual RPM3 Units can be clustered into Sites. RPM3s can only communicate with other RPM3s that have matching Unit and Site assignments. There are 16 Unit addresses that range between 0 and 16. Sites are addressed 0 to 7. Unit and Site addresses are stored in EEPROM. To allow the use of multipoint services, the Unit address can be altered without writing the new address to EEPROM using the ADDR command.

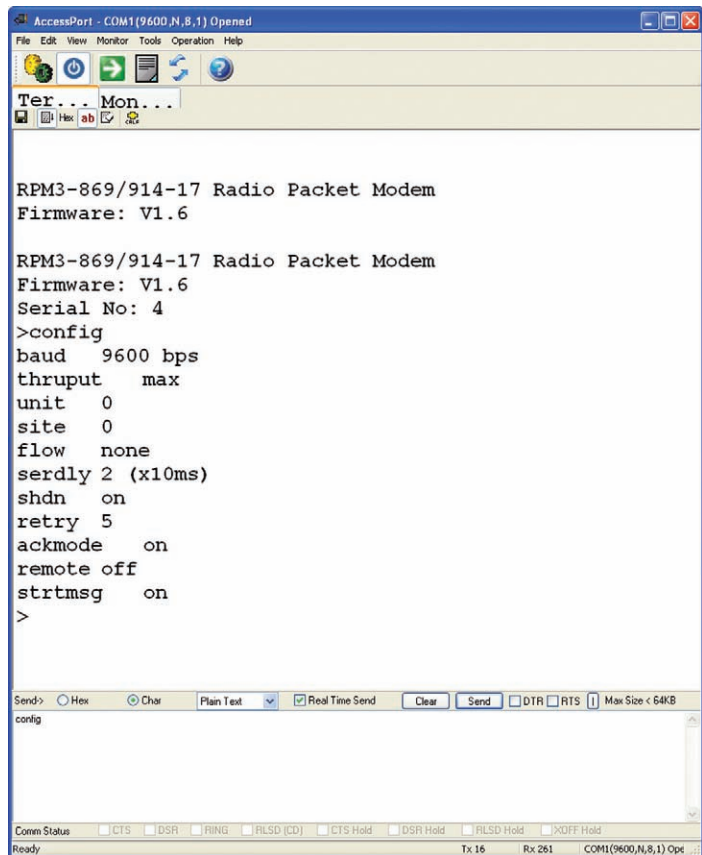
Flow control is rather obvious here. However, the SERDLY (Serial Delay) setting is the amount of time given to an internal timer to allow the full transmission of the message in the transmit buffer if the buffer is not full.

SHDN set to ON activates the WAKE/DTR function. With SHDN (Shutdown) set to ON, we can control the RPM3's power consumption by entering and exiting sleep mode with the DTR pin. Recall that PIC18F4550's DTR and the WAKE I/O lines are the same I/O pin.

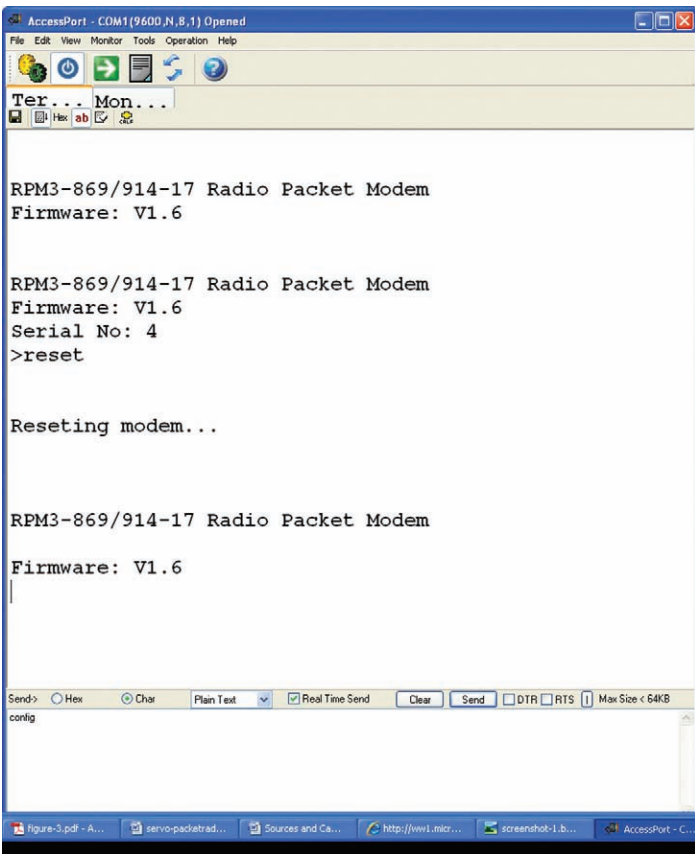
The RETRY, ACKMODE, STRTMSG, and



SCREENSHOT 2. We know we've entered configuration mode when the RPM3's serial number is displayed.



SCREENSHOT 3. These are the default settings. Issuing the CONFIG command also sets the RPM3's UART for three-wire mode (Tx-Rx-Gnd).



SCREENSHOT 4. The RPM3 was reset by the PIC18F4550 in this capture. Any command that can be manually directed to the RPM3 can be issued electronically, as well.

REMOTE entries are self-explanatory. You can retry a transmission up to 63 times. However, we can't remotely configure a radio at this time.

Let's say we've made our changes and need to put them into effect with a PIC18F4550-generated reset. In fact, we could have entered configuration mode in a similar manner using the PIC18F4550's USART. Here's how we talk to the RPM3 programmatically:

```
if(rloopcounter1 == 0 && counter == 2)
{
    RS232_Out_Data[0] = 'R';
    RS232_Out_Data[1] = 'E';
    RS232_Out_Data[2] = 'S';
}
```

```
RS232_Out_Data[3] = 'E';
RS232_Out_Data[4] = 'T';
RS232_Out_Data[5] = 0x0D;
RS232_Out_Data[6] = 0x00;
LastRS232Out = 6;
RS232_Out_Data_Rdy = 1;
RS232cp = 0;
rSETUP = 1;
counter = 3;
}
```

I think the idea here is crystal clear. Note that we pulled the RPM3's SETUP pin logically high before the reset completed. Otherwise, the RPM3 would revert right back into configuration mode. The whole process from reset to setup to reset is captured in **Screenshot 4**. Once we're back in connected mode, sending and receiving data is as easy as stuffing the RS232_Out_Data buffer and reading the USB_Out_Buffer. If you don't want the incoming data streamed out of the PIC18F4550's USB port, you will have to turn off the CDC task to keep the incoming data from being automatically gleaned from the USB_Out_Buffer.

Applications a Plenty

I've shown you how to use the RPM3 with a PIC18F4550. However, you can apply the RPM3 in a stand-alone environment with just about any microcontroller that can signal using RS-232. The ability to address a single RPM3 or a herd of RPM3s under microcontroller control makes this radio perfect for entry monitoring, robotic telemetry, and data logging. You can even issue a RPM3 to every steel soldier that needs to communicate with its peers or the aluminum admiral. By the way, the +5.0 volt RPM3 WILL NOT run at +3.3 volts. **SV**

www.servomagazine.com/index.php?/magazine/article/february2011_Eady

Fred Eady can be reached via email at fred@edtp.com.

Sources

Lemos International
RPM3
www.lemosint.com

EDTP Electronics, Inc.
Emperor/PIC18F4550
www.edtp.com



Das Blinkenboard

Not just for blinken LEDs.

Much Much More!

<http://store.nutsvolts.com> & <http://store.servomagazine.com>



Playstation Servomotor Controller Interface

As of 2009, the PS2 has sold over 140 million units. With the PS2 still in production and so many controllers available, it's a natural to apply this controller for more than games. This makes it an excellent controller for servomotors and robotics.

The PS2 SMC (Servo Motor Controller) interface allows you to control six hobby servomotors using the Playstation controller (also called the Dual Shock 2 [DS2]). Both corded and cordless versions are supported. Therefore, the wireless SMC can be used for remote control of your moving platform.

by John Iovine

The PS2's analog sticks give precise servomotor movements with speed control. Rumble feedback occurs when the servomotor is instructed to move out of range (stalling).

Sony has not released the communication protocol of its PlayStation controller, however, there are a few Internet sites that have teased out enough of the details for us to work with.

The technical specifications I used were from:

- mikroElektronika
www.mikroe.com/forum/viewtopic.php?t=8792
- *Nuts & Volts* September '03 column
by Jon Williams
- Curious Inventor
<http://store.curiousinventor.com/guides/PS2>

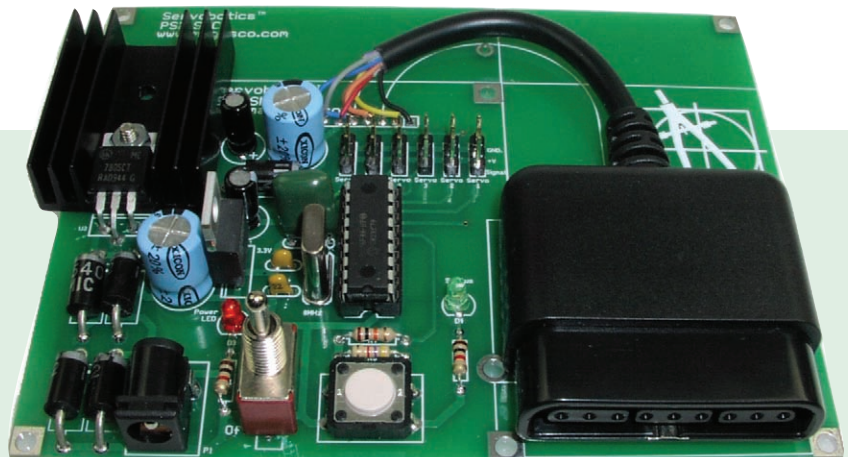


FIGURE 1. The PS2 interface controller.

I will not repeat the communication protocol information available on these sites, except to discuss how it relates to our current project. If you want to dig deeper into the PlayStation controllers communication, you can use those references cited.



FIGURE 2. PS2 with wireless Playstation controller connected to six servos.

Construction

The PS2 controller has a nine-pin connector shown in **Figure 3**. If you are hardwiring this circuit from scratch, you need to obtain a PS2 extension cable. Leave about 4-5 inches of cable attached to the female connector also shown in **Figure 3**. Cut and strip the wires inside the cable to attach to your circuit.

Wire definitions are: 1-Brown is Data; 2-Orange is Command; 3-Grey is Vibration; 4-Black is Ground; 5-Red is Power; 6-Yellow is Attention; 7-Blue is Clock; 8-White is

Unknown; and 9-Green is Acknowledge.

We use seven of the nine wires off the connector with the exceptions being the green and white wires. These are fine wires which are a little difficult to strip and solder. My fingers are a little clumsy for this detail work, so I prefer to use a pre-made PS2 interface cable with a seven-position socket that plugs onto a header. The schematic for our circuit is shown in **Figure 4**. You can hardwire and build this circuit on a prototyping breadboard (see **Figure 5**) or purchase the kit.

To build the kit, we start with the PCB (**Figure 6**). The kit is available in three versions that have different power supply ratings of either one ampere, three ampere, or five ampere. The higher amps can supply more power to the servomotors without overheating.

Begin construction by mounting components and parts on the silkscreen side of the PCB. R1 and R2 are 1K resistors (color bands are brown, black, red). R3 is a 4.7K resistor (color bands are yellow, purple, red); R4 is a 10K resistor (color bands are brown, black, orange).

Check this out in **Figure 7**.

Next, mount and solder the ICS-18 microcontroller socket labeled U2. Then, mount and solder the 1N5402X diodes labeled D3-D7. Keep the line on the diode orientated to the line of the silkscreen outline of the diode. Next, mount the two colored LEDs. The red LED is labeled Power LED; the green LED is labeled Status LED. Orientate the LEDs properly. The longer lead of the LED is positive. This faces the round portion of the LED outline. The shorter lead of the LED is negative; it faces the flat side of the LED silkscreen outline as shown in **Figure 8**.

Now, mount and solder the round pushbutton switch

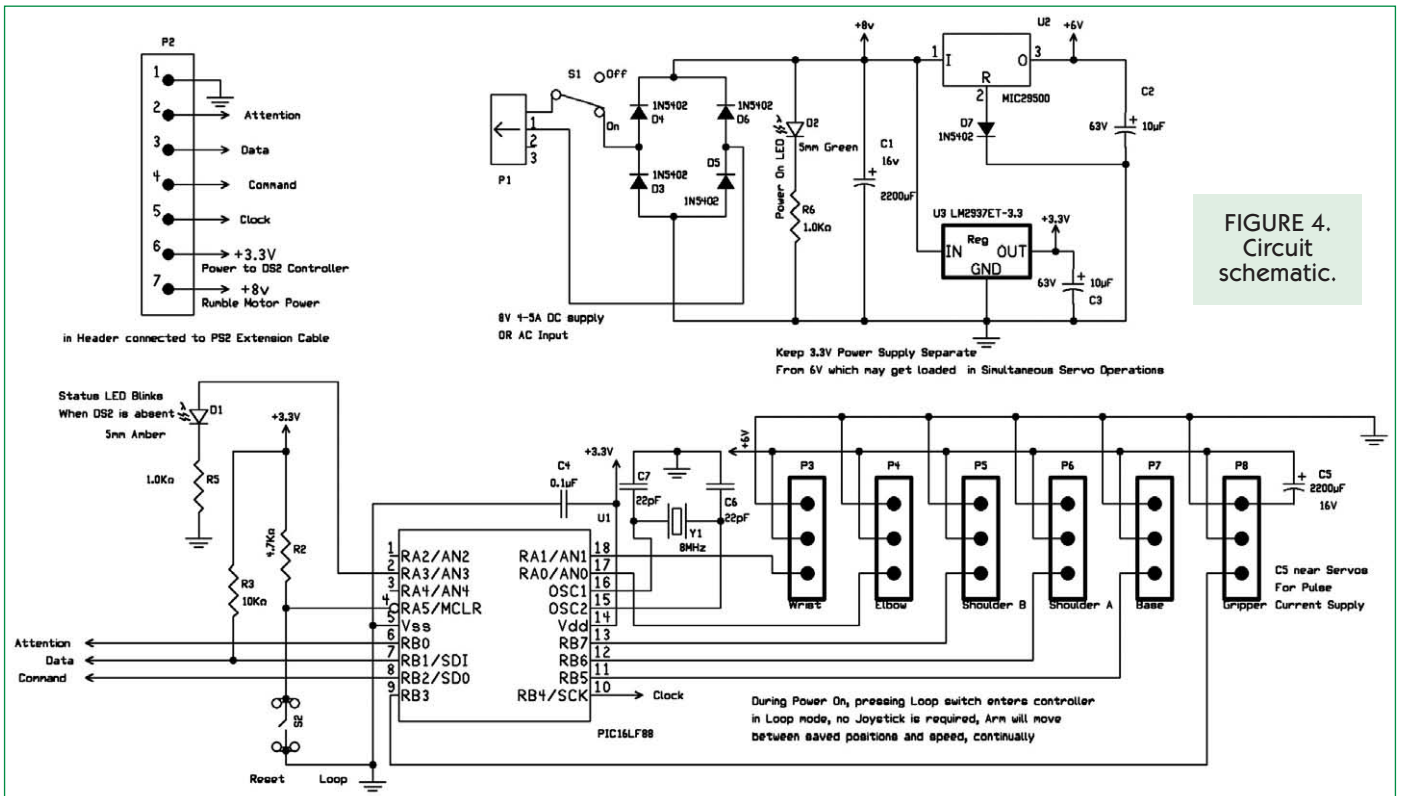


FIGURE 4. Circuit schematic.

* Depending on which version of the PS2 controller you purchased, this regulator may be changed to the three amp or five amp version.

and the toggle switch labeled S1. Mount and solder the power supply jack labeled P1. When mounting capacitors, the longer lead is the positive lead. Capacitors C2 and C6 are 1,000 μ f. C5 and C7 are 10 μ f capacitors. The following capacitors are non-polarized and may go in with the leads facing either way. Capacitor C4 is a 0.1 μ f 100V. C1 and C3 are small 22 pF capacitors. Next, mount and solder the 8 MHz crystal labeled Y1. Now, mount the SMH-03's labeled P2-P7, then mount the seven-pin header P8 and the LM2937 labeled U1. Before mounting the 7805* voltage regulator, first attach it to the heatsink (heatsink is included with the three and five amp versions only) and then mount it to the PCB in the position labeled U3 (see **Figure 9**).

When mounting voltage regulators LM2937 and 7805* to the PCB, the flat side must be aligned with the small rectangle printed on the board.

Plug the PS2 connector into the P8 connector. The black wire of the connector faces toward the right, with the PS2-SMC-06 facing you. Secure the PS2 connector to the top side of the PCB as in **Figure 10**.

The hex file for programming the 16F88 accompanies this article. If you purchased the kit, the PIC microcontroller is pre-programmed.

Power Supply

The PS2 SMC can use a variety of power supplies from 6V to 9V (AC/DC, one Amp, or more) thanks to the onboard rectifiers. You don't have to worry about DC supply polarity. On my prototype, power is supplied to the board through the 2.1 mm DC barrel jack. The circuit can be reset anytime using a momentary contact reset switch.

PS2 SMC Operation

Begin by plugging in nine-pin male connector of your PS2 controller or wireless receiver into the female connector on the circuit board.

Servomotor Connectors

There are six standard three-pin male headers for connecting hobby servomotors Servomotor 1 to Servomotor 6. Connect one to six servomotors as shown in **Figure 11**.

Power-Up

Plug in the PS2 controller, connect the servomotors, and toggle the on/off switch to the "on" position. The green LED will light indicating that power is on. The amber LED (status LED will go on after a couple of seconds) indicates a communication link has been established successfully with the PS2 and that the servomotors are now ready to be controlled. This amber LED will stay on. At the same time, the red LED on your PS2 will go on, indicating

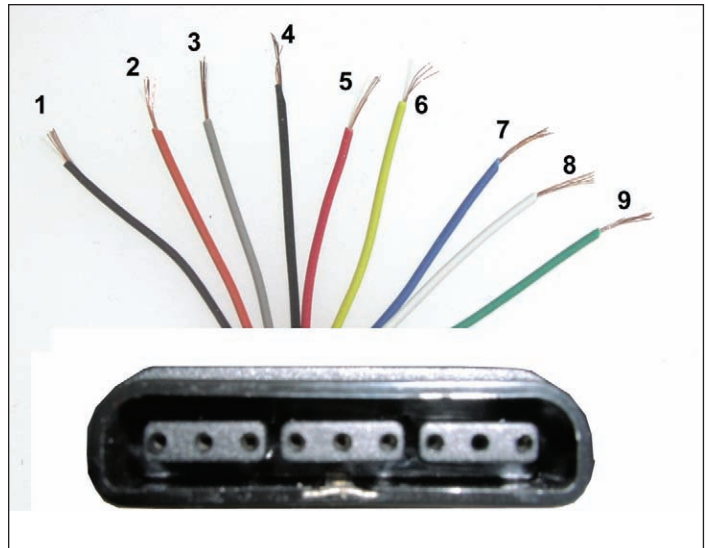


FIGURE 3. Playstation nine-pin connector with wires out.

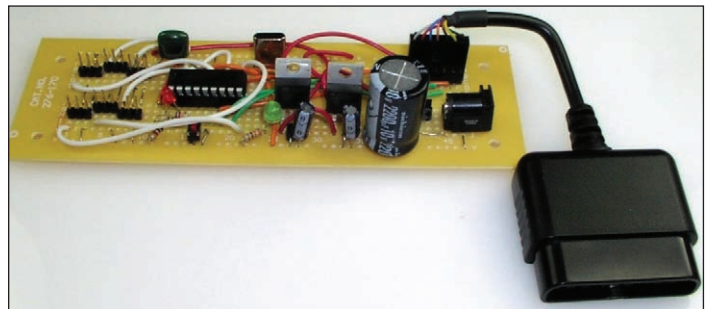


FIGURE 5. Prototype PS2 interface controller.

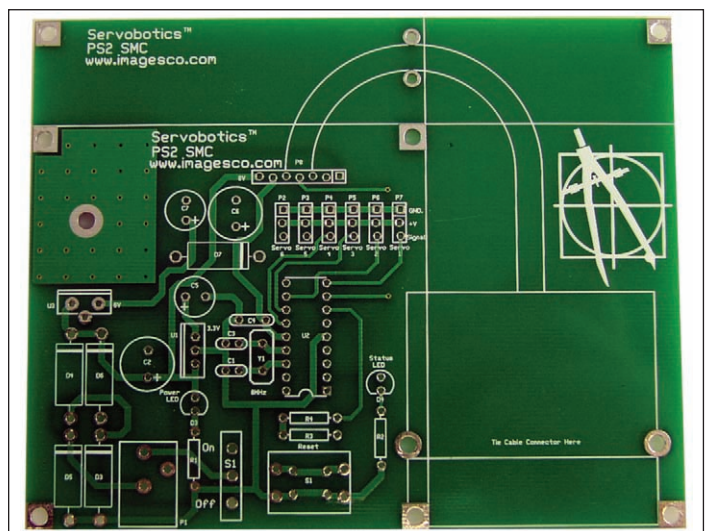


FIGURE 6. PCB for interface controller.

the PS2 is in analog mode. All six servomotors in normal Independent mode will move to the center neutral position on power-up.

When using the cordless PS2, make sure the communication link between the PS2 and its wireless receiver is established. Pressing any of the action buttons on the PS2 will help establish the wireless link on power-up. When a proper link between the PS2 and its wireless

FIGURES 7-10. PCB construction.

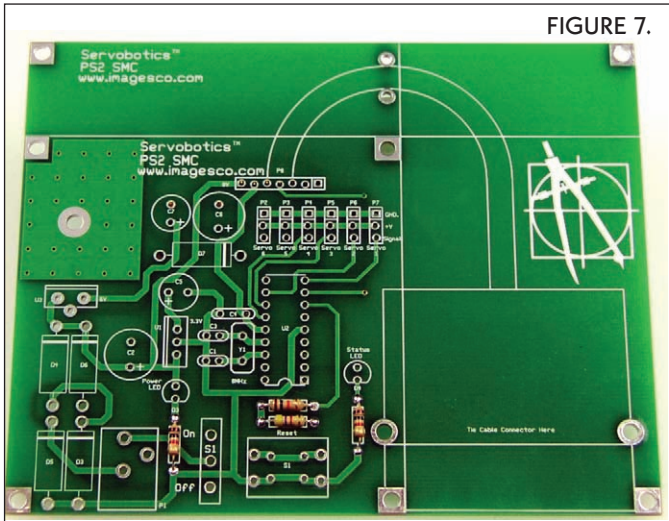


FIGURE 7.

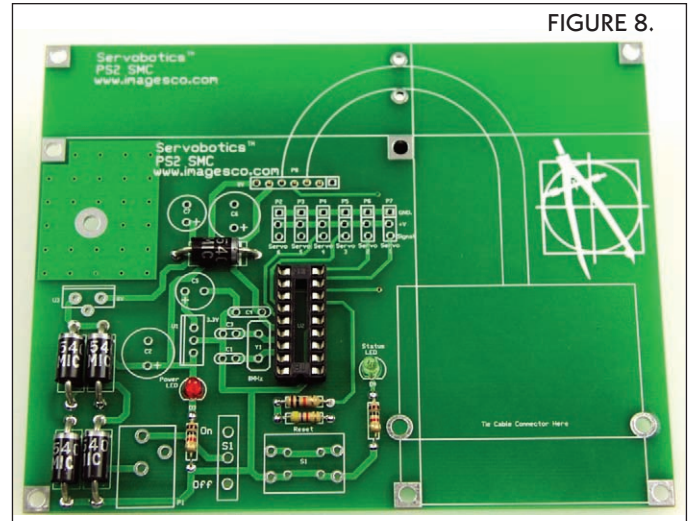


FIGURE 8.

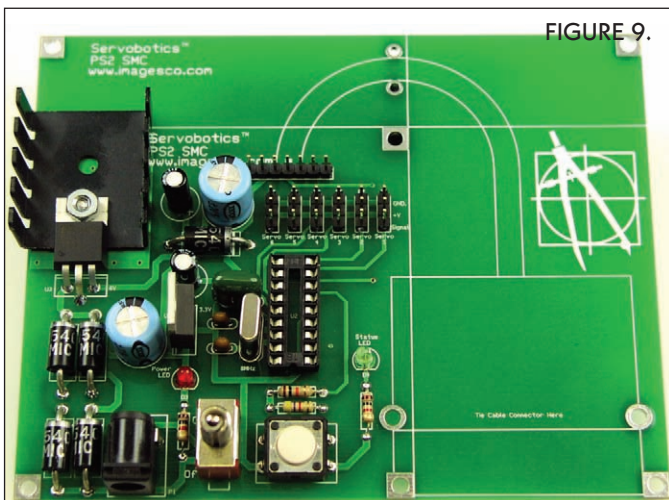


FIGURE 9.

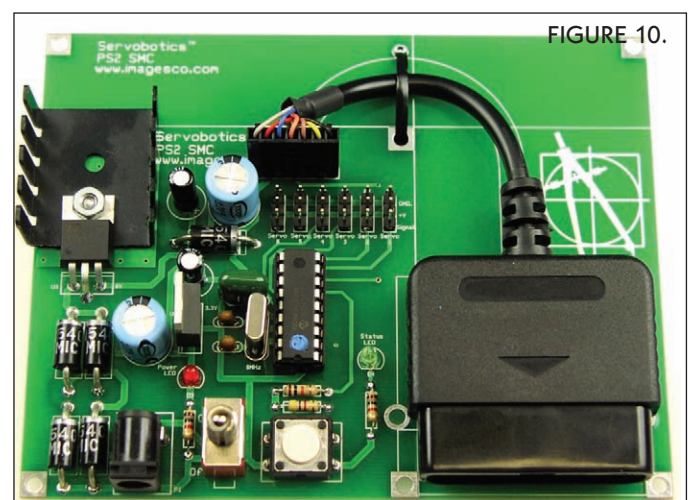


FIGURE 10.

receiver is established, pressing the action button will result in the LED blinking on the wireless receiver.

Controller Map

The six servomotors are controlled by the two analog

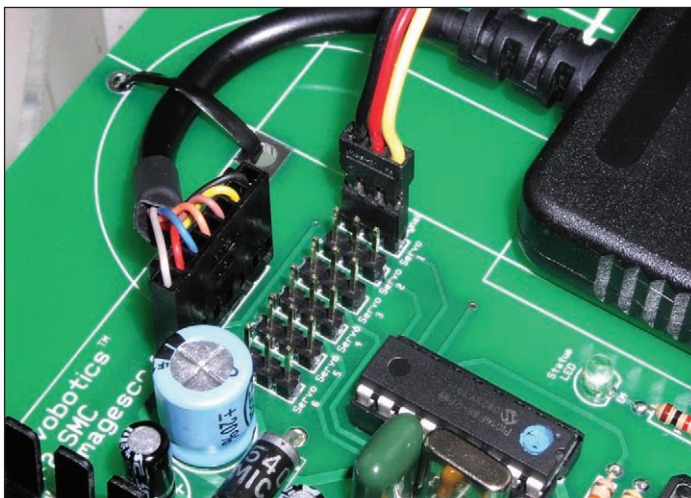


FIGURE 11. Servomotor connections.

joysticks (thumb sticks) on the PS2 controller; see **Figure 12**. Servomotor 1 and Servomotor 2 are controlled by the left analog joystick. The X axis of the left analog joystick controls Servomotor 1 and the Y axis controls Servomotor 2. Tilting the joystick along any axis will result in motion of the servomotor associated with that axis. Tilting the joystick in the opposite direction along the same axis will result in servomotor motion in the opposite direction.

Servomotor speed is proportional to the tilt angle of the joystick. Tilting the joystick slightly will result in the slowest servomotor motion. Tilting the joystick all the way will result in fastest servomotor motion. There are three levels of servomotor speed along one direction of the axis.

Releasing the joysticks will bring it to center position and the servo will stop at its current position.

Since sticks may not always return to the center neutral position, there is a dead band around the center of the stick which results in no movement. This ensures the servomotor doesn't keep moving in case the stick has not returned to the neutral position.

Servomotor 3 and Servomotor 4 are controlled by the right analog stick in the same manner as described for the left analog stick. Pressing Left 1 (shoulder button) in front of the PS2 will shift control of the right analog stick from

Servomotor 3 – Servomotor 4 to Servomotor 5 – Servomotor 6. Servomotor 5 and 6 are controlled by the right analog stick as long as the left 1 button is pressed. Releasing the left 1 button will shift control of the right analog stick back to Servomotor 3 and 4.

Rumble

The servomotor has a maximum limit in both directions and driving it beyond these limits will cause damage to the servomotor and drain heavy current from the power supply. To avoid this, servomotors are not allowed to be driven beyond their Pulse Width (PW) limits, as determined by the manufacturer's recommendation for PW signals sent to the servomotor. When any of the servomotors have reached their limit and the user continues trying to move it by tilting the corresponding analog stick, the servomotor will not move and this condition will be indicated by a Rumble (force feedback) feature. Rumble will stay on as long as the user keeps the stick tilted.

Programmable Configuration Modes

The PS2 SMC can be configured in three different operating modes. These modes can be changed "on-the-fly" during normal operation. New mode selection is stored in the non-volatile memory of the PIC. The stored mode will automatically load the next time the PS2 SMC is turned on.

1.) Mobile Platform Mode – MPM (Circle and Square Buttons)

In this mode, Servomotor 1 and Servomotor 2 are configured as differential drive in a moving platform (for continuous rotation servomotors). To activate the MPM, press the Circle button.

Pressing the Square button will select the Normal Independent Mode (NIM) for servomotor control. This mode is for standard servomotors and not continuous rotation ones.

In MPM mode, the left analog stick will act as a navigation control with built-in speed control. Tilting the stick upwards will result in forward motion; downwards is for reverse motion; left is for left turn; and right is for right

Press Left Button for Servo's 5 & 6



FIGURE 12. Servomotors mapped to PlayStation controller.

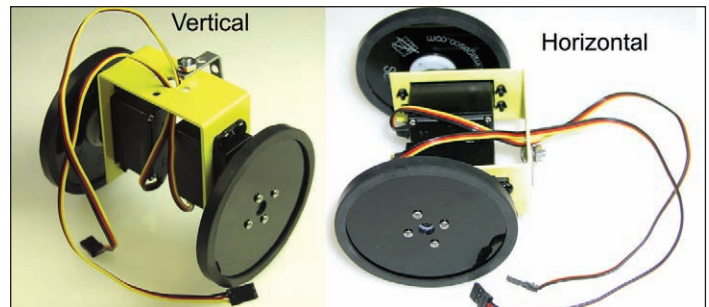


FIGURE 13. Mobile test platform.

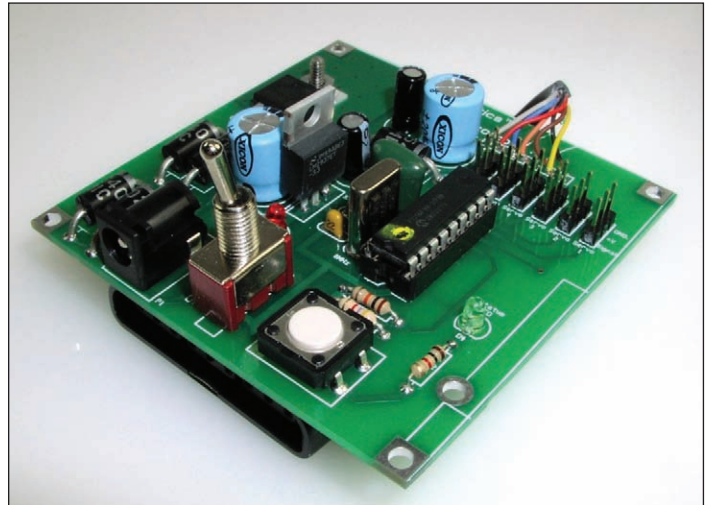


FIGURE 14. Reduced size PCB for mobile operations on robots.

Parts List

QTY	ITEM
(1)	Printed Circuit Board
(1)	ICS-18
(5)	1N5401-1N5402
(1)	Sub-min LED
(1)	Green LED
(6)	Three-Pin Straight Male Header
(1)	Seven-Pin Straight Male Header
(1)	8.0 MHz Crystal
(1)	7805 (Voltage Regulator)
(1)	HeatSink-03
(2)	1K ohm 1/4 watt Resistor
(2)	4.7K ohm 1/4 watt Resistor
(1)	10K ohm 1/4 watt Resistor
(1)	2.5 mm PC Power Jack Connector (PJ-102B)

(1)	12 mm Momentary Switch Tact (SW-25)
(1)	Toggle Switch SPDT On-On PCM (SW-07)
(2)	10 μ F Capacitor, 16V
(1)	0.1 μ F Capacitor, 100V
(2)	22 pF Capacitor, 50V
(2)	1,000 μ F Capacitor, 10V
(1)	LM2937
(1)	PIC16F88 Microcontroller
(1)	Seven-Position Modular to CS Connector (Cable-PS2)
(1)	Jack-09
(1)	9 VDC 300 mA. (or greater) Power Supply

A complete or partial kit to go with this article can be purchased online from the SERVO Webstore www.servomagazine.com or call our order desk at 800-783-4624.

Technical Supplement PS2

PS2 Controller

Before going further, we need to first look into the details of what the SPI interface will be communicating to the host. This involves the PS2 control buttons and analog joysticks.

The PS2 has 16 buttons: four Directional Pad (Dpad) buttons; four Action buttons (circle, triangle, etc.); and four Shoulder buttons, (start, select etc.). Both analog joysticks can be pressed as buttons, in turn. When pressed, each button outputs -0; when released each button outputs -1.

The two analog joysticks on the PS2 can each be moved in two axis. The output of each stick is proportional to the amount of tilt the stick is placed under. Each stick outputs two values of 0 to 255, corresponding and proportional to the tilt on each axis.

Twelve of the PS2 button may be set as Pressure Sensitive buttons (four Dpad, four Action, four Shoulder), each outputting a value between 0 to 255, proportional to the amount of pressure exerted on the buttons.

We can mix and match the function of the buttons and sticks depending on which of the PS2 three modes of operation has been set up (either digital, analog, or analog pressure).

In digital mode, analog sticks don't output a value; pressure values are also not present. Only the on/off state of the digital buttons is outputted.

In analog mode, the digital on/off state is outputted, as well as the analog joystick values.

In analog pressure mode, digital on/off, analog sticks, and pressure values all are outputted.

The SPI to the host is a synchronous serial half/full duplex communication protocol. The PS2 controller is the slave and the host controller — in our case, the PIC microcontroller — is the master; and generates synchronizing clock. The master PIC sends command bytes to the PS2, and the PS2 sends data bytes back (full duplex mode).

Commands have a three-byte header. These bytes request the PS2 to acknowledge its current mode of operation. The PS2 responds with three data bytes indicating its current mode.

The current state of the buttons, stick, and pressure values can be requested from the PS2 by a polling method where a specific button value's command is sent and the PS2 responds with the current state.

Every command is divided into a sequence of bytes. The header is followed by data. The digital button's polling command, for example, is five bytes. So, to know button states, the master has to transmit five command bytes to the PS2; the PS2 will return five data bytes back to the master. In contrast, the analog polling commands consist of nine bytes.

In order to select the mode of operation, turn on/off the vibration motors the PS2 needs to be configured (with configuration commands).

PIC Program

The general function of the PIC program is as follows:

When power is applied, the PS2 default is the digital mode. The PIC program sends a wireless idle command to the PS2 that is specifically used for wireless. The command consists of five bytes

and data returned is not used.

Commands to the controller should be separated by five to 10 msecs. However, there is no delay between bytes used in a command.

Next, the digital button's polling command of five bytes is sent to the PS2, for which the PS2 returns the button states in packets of five bytes. Here, the second data byte from the PS2 is a mode of operation byte. This byte should be \$41 hex. If it isn't, either the PS2 is absent or is faulty. So the mode byte from the PS2 is used as a reference and benchmark for proper communication. This command is only to make sure the PS2 is up and working.

Next, the PS2 should be configured for the proper mode of operation. For that, the configuration enter command is given to the PS2.

Once into the configuration setup, subsequent commands such as Vibration Motors On, Pressure Values On, and Analog Lock Mode are given to the PS2 sequentially. Once the configuration is set up to our requirements, the exit command is given to the PS2.

Here are a few things you should know:

1. In the configuration mode, if the PS2 is not sent a command for some time, it automatically falls out of configuration mode.
2. The configuration enter command length is equal to that of the current mode of operation. For example, if the current mode is digital, the configuration enter command will have the same number of bytes as that of the digital polling command.
3. Once into configuration mode, all configuration commands (including configuration exit commands) are nine bytes in length.
4. The PS2 will output the current mode in a data string (second byte) in response to a configuration enter command. After that, it will output configuration mode bytes in data strings.

After configuration, a command is again set to the required length for the analog - pressure poll command i.e., 21 bytes and various command bytes are initialized as per the requirements. Next, comes the servomotor control loop. Every 20 msecs, the PS2 is polled by the analog pressure poll command, and a new servomotor value is calculated based on current button/stick states; new values are pulsed out.

We don't use pressure-sensitive buttons in our program.

For each command, every byte has to be per the specifications. You will notice that for each command, command bytes are initialized, the byte length of a command is set, and then the SPI subroutine will be called. The SPI subroutine will return data bytes.

If after configuration, the PS2 doesn't function, it has to be reconfigured again. So, in the servo loop, the second data byte (mode byte) is checked every time the PS2 is polled. If that byte is not what we expect it to be, the PS2 is reconfigured again. The PS2 also needs to be polled for a minimum number of times in one second period. Failing to do so will result in the PS2 going back into the default state.

We are able to maintain a 125 kHz clock frequency for the synchronous clock in SPI between the PIC and PS2.

General Notes

1. One should plug the PS2 controller and servomotors on the interface board before powering up. Only remove them after turning off the board.
2. Configuration modes can be changed anytime the user wants in normal operation. However, since the associated mechanical hardware connected to the motors for different modes is so different, practically speaking, a need for changing from one mode to another won't arise during an operation. Mode change can be done when required immediately at power-on, once the status LED goes on. This will ensure that the servomotors are at the center position when the mode is changed so an unwanted movement of the servomotors won't occur as a result of mode change.
3. Servomotor control is inhibited when the mode change buttons are pressed.

turn of your moving platform. The tilt angle of the stick will determine the speed. Two-level speed control is implemented.

Sometimes the stick doesn't return to the neutral position on release which may result in unwanted movements of the servomotors. Creating a small dead space or band around the center position of the joystick eliminates this movement. Within this dead band, the servomotors aren't activated. To test this mode, I created a quick and dirty mobile platform using two continuous rotation (CR) servomotors and a dual servomotor bracket (see **Figure 13**).

For mobile operations or if you're just tight on space, the PCB can be cut down (before assembly) so its footprint is much smaller (**Figure 14**). The PS2 connector is secured

to the bottom of the PCB.

2. Opposite Phase Mode – OPM (Cross and Triangle Buttons)

Servomotor 3 and Servomotor 4 can be configured to be controlled independently or in Opposite Phase Mode (OPM). In OPM, each servomotor shaft moves in the opposite direction from the other, 180 degrees out of phase. Pressing the Cross button will select OPM mode. Pressing the Triangle button will select Normal Independent Mode (NIM).

OPM can be helpful when two servomotors are connected to a common load, or when you're using two servomotors for increasing the overall relative speed of movement.

In OPM, the X axis of the right analog stick will control Servomotor 3 and 4. Tilting the stick along the X axis in one direction will result in the simultaneous opposite movement of Servomotor 3 and Servomotor 4 at the same speed.

Tilting the stick in the opposite direction will result in the simultaneous opposite movement of Servomotor 3 and 4, but in the opposite direction to that of the previous case. As in MPM, there is a dead band around the center of the stick and three-level speed control proportional to the stick tilt.

Servomotor 5 and Servomotor 6 always operate in NIM.

3.) Normal Independent Mode – NIM (Square and Circle Buttons)

Pressing the Square button will select NIM for servomotor control. This mode is for standard servomotors and not CR servomotors.

Handiness (Righties and Lefties)

When selected, MPM mode by default sets the left analog stick to control Navigation. For righties, they can switch Navigation control to the right analog stick by pressing the right arrow button on the directional pad. Pressing the left arrow button on the directional pad brings the stick controls back to their default mapping. In Righty mode, the left analog stick will control Servomotor 3

and Servomotor 4; the right stick controls Servomotor 5 and Servomotor 6.

When the Shoulder button is pressed, the right analog stick will control Servomotor 1 and Servomotor 2. Basically, Leftie and Righty mode are independent of other modes selected; control between the two sticks is swapped, irrespective of the mode.

Wrap-Up Mode

Utilizing existing equipment in your robot projects is always satisfying. Using the PS2 controller and joysticks adds to the enjoyment of deploying your builds. **SV**



**MOTORS
GEARBOXES
WHEELS
AND MORE**

BB BaneBots **BANEBOTS.COM**
970-461-8880

Zen happens when you achieve enlightenment.
Emphasis on the 'light' part. Light powers the coil that rolls the ball.
That's how Zen really works, you know.



The Zendulum is a simple kit with clever electronics. Give it light, and watch the ball roll back and forth - check it out online to see the ultimate desk toy in action!

SOLARBOTICS Ltd
www.solarbotics.com 1-866-276-2687

PS- The kit has resistors, so as you assemble it you can chant "Ohmmmm..."



The *SERVO* Webstore

Attention Subscribers ask about your discount on prices marked with an *

CD-ROM SPECIALS



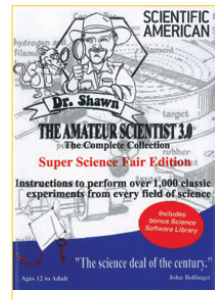
6 CD-ROMs & Hat Special
Only \$ 129.95 or \$24.95 each.
www.servomagazine.com

The Amateur Scientist 3.0 The Complete Collection

by Bright Science, LLC

There are 1,000 projects on this CD, not to mention the additional technical info and bonus features. It doesn't matter if you're a complete novice looking to do your first science fair project or a super tech-head gadget freak; there are enough projects on the single CD-ROM to keep you and 50 of your friends busy for a lifetime!

Reg \$26.95 Sale Price \$23.95



ROBOTICS

PIC Robotics

by John Iovine

Here's everything the robotics hobbyist needs to harness the power of the PICMicro MCU!

In this heavily-illustrated resource, author John Iovine provides plans and complete parts lists for 11 easy-to-build robots each with a PICMicro "brain." The expertly written coverage of the PIC Basic Computer makes programming a snap – and lots of fun.

\$24.95



FIRST Robots: Rack 'N' Roll: Behind the Design

by Vince Wilczynski,
Stephanie Slezyski

More than 750 photographs!

The second annual book highlighting the creativity and process behind 30 winning robot designs from the 18th annual international FIRST Robotics Competition. The FIRST organization, founded by Dean Kamen (inventor of the Segway), promotes education in the sciences, technology, and engineering.

Reg \$39.95

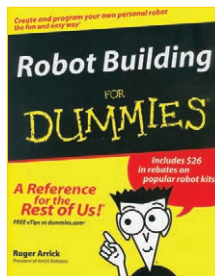


Robot Building for Dummies

by Roger Arrick / Nancy Stevenson

Discover what robots can do and how they work. Find out how to build your own robot and program it to perform tasks. Ready to enter the robot world? This book is your passport! It walks you through building your very own little metal assistant from a kit, dressing it up, giving it a brain, programming it to do things, even making it talk. Along the way, you'll gather some tidbits about robot history, enthusiasts' groups, and more.

\$24.95



Build Your Own Humanoid Robots

by Karl Williams

GREAT 'DROIDS, INDEED!

This unique guide to sophisticated robotics projects brings humanoid robot construction home to the hobbyist. Written by a well-known figure in the robotics community, *Build Your Own Humanoid Robots* provides step-by-step directions for six exciting projects, each costing less than \$300. Together, they form the essential ingredients for making your own humanoid robot. **\$24.95***



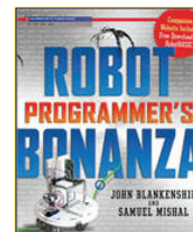
Robot Programmer's Bonanza

by John Blankenship,
Samuel Mishal

The first hands-on programming guide for today's robot hobbyist!

Get ready to reach into your programming toolbox and control a robot like never before! *Robot Programmer's Bonanza* is the one-stop guide for everyone from robot novices to advanced hobbyists who are ready to go beyond just building robots and start programming them to perform useful tasks.

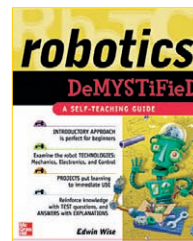
\$29.95



Robotics Demystified

by Edwin Wise

YOU DON'T NEED ARTIFICIAL INTELLIGENCE TO LEARN ROBOTICS! Now anyone with an interest in robotics can gain a deeper understanding – without formal training, unlimited time, or a genius IQ. In *Robotics Demystified*, expert robot builder and author Edwin Wise provides an effective and totally painless way to learn about the technologies used to build robots! **\$19.95**



**We accept VISA, MC, AMEX,
and DISCOVER**
Prices do not include shipping and
may be subject to change.

To order call 1-800-783-4624

SERVO Magazine Bundles



Published by T & L Publications, Inc.

\$57
per bundle

Save \$10
off the
normal
price!!

Now you can get one year's worth of all your favorite articles from *SERVO Magazine* in a convenient bundle of print copies. Available for years 04, 05, 06, 07, 08, and 09.

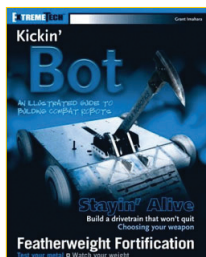
Kickin' Bot

by Grant Imahara

Enter the arena of the metal gladiators!

Do you have what it takes to build a battle-ready robot? You do now! Here are the plans, step-by-step directions, and expert advice that will put you in competition — while you have a heck of a lot of fun getting there. Grant Imahara, the creator of the popular BattleBot Deadblow, shares everything he's learned about robot design, tools, and techniques for metal working and the parts you need and where to get them.

\$24.95

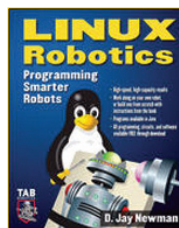


Linux Robotics

by D. Jay Newman

If you want your robot to have more brains than microcontrollers can deliver — if you want a truly intelligent, high-capability robot — everything you need is right here. *Linux Robotics* gives you step-by-step directions for "Zeppo," a super-smart, single-board-powered robot that can be built by any hobbyist. You also get complete instructions for incorporating Linux single boards into your own unique robotic designs. No programming experience is required. This book includes access to all the downloadable programs you need.

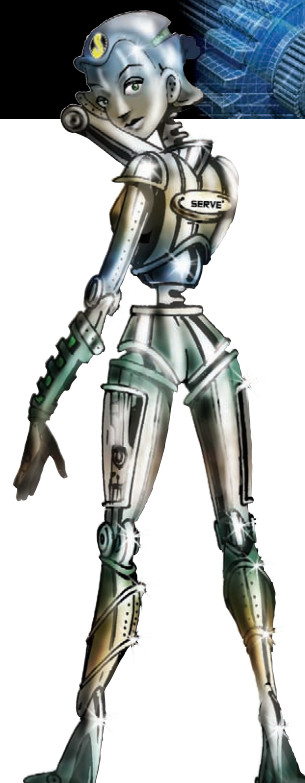
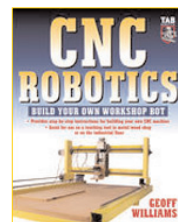
\$34.95



CNC Robotics

by Geoff Williams

Here's the FIRST book to offer step-by-step guidelines that walk the reader through the entire process of building a CNC (Computer Numerical Control) machine from start to finish. Using inexpensive, off-the-shelf parts, readers can build CNC machines with true industrial shop applications such as machining, routing, and cutting — at a fraction of what it would cost to purchase one. Great for anyone who wants to automate a task in their home shop or small business. **\$34.95**



SPECIAL OFFERS

RobotBASIC Projects For Beginners

by John Blankenship, Samuel Mishal

If you want to learn how to program, this is the book for you. Most texts on programming offer dry, boring examples that are difficult to follow. In this book, a wide variety of interesting and relevant subjects are explored using a problem-solving methodology that develops logical thinking skills while making learning fun. RobotBASIC is an easy-to-use computer language available for any Windows-based PC and is used throughout the text.

Reg. Price \$14.95 Sale Price \$9.95



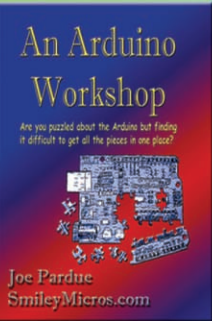
Technology Education Package for Everyone Starting in Electronics

This lab — from the good people at GSS Tech Ed — will show you 40 of the most simple and interesting experiments and lessons you have ever seen on a solderless circuit board. As you do each experiment, you learn how basic components work in a circuit. Along with the purchase of the lab, you will receive a special password to access the fantastic online interactive software to help you fully understand all the electronic principles. For a complete product description and sample software, please visit our webstore.

Regular Price \$79.95

Subscriber's Price \$75.95

SPECIAL OFFERS

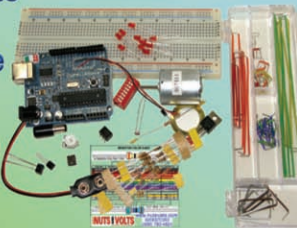


An Arduino Workshop
Are you puzzled about the Arduino but finding it difficult to get all the pieces in one place?
Joe Pardue
SmileyMicros.com
Book \$44.95

Puzzled by the Arduino?

Based on the *Nuts & Volts* Smileys Workshop, this set gives you all the pieces you need!

Book and Kit Combo
\$124.95



Kit \$84.95

For more info on this and other great combos, please visit: <http://store.nutsvolts.com>



Getting Started with PIC's
A Collection of 2008 Nuts & Volts Magazine Articles
Chuck Hellebray

Enter the world of PICs & Programming with this great combo!

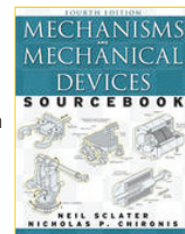



Combo Price
\$175.95

For complete details visit our webstore @ www.nutsvolts.com

Mechanisms and Mechanical Devices Sourcebook

by Neil Sclater, Nicholas Chironis
Over 2,000 drawings make this sourcebook a gold mine of information for learning and innovating in mechanical design. Overviews of robotics, rapid prototyping, MEMS, and nanotechnology will get you up to speed on these cutting-edge technologies. Easy-to-read tutorial chapters on the basics of mechanisms and motion control will introduce those subjects to you. **Reg \$89.95 Sale Price \$69.95**



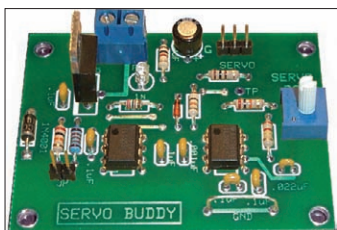
Forbidden LEGO

by Ulrik Pilegaard / Mike Dooley
Forbidden LEGO introduces you to the type of free-style building that LEGO's master builders do for fun in the back room. Using LEGO bricks in combination with common household materials (from rubber bands and glue to plastic spoons and ping-pong balls) along with some very unorthodox building techniques, you'll learn to create working models that LEGO would never endorse. **Reg \$24.95 Sale Price \$19.95**



PROJECTS

The SERVO Buddy Kit



An inexpensive circuit you can build to control a servo without a microcontroller.



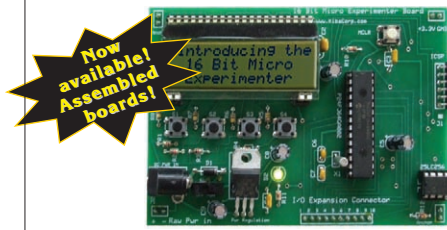
For more information, please check out the May 2008 issue or go to the **SERVO** website.

Includes an article reprint.

Subscriber's Price **\$39.55**

Non-Subscriber's Price **\$43.95**

16-Bit Micro Experimenter Board



Ready to move on from eight-bit to 16-bit microcontrollers? Well, you're in luck!

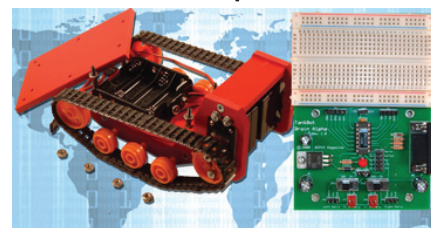
In the December 2009 *Nuts & Volts* issue, you're introduced to the 16-Bit Micro Experimenter.

The kit comes with a CD-ROM that contains details on assembly, operation, as well as an assortment of ready-made applications. New applications will be added in upcoming months.

Subscriber's Price **\$55.95**

Non-Subscriber's Price **\$59.95**

Tankbot Kit & Brain Alpha Kit



Tankbot/Brain Alpha

originally by Ron Hackett

A series filled with projects and experiments to challenge you through your learning process while you grow your fully expandable Brain Alpha PCB! The brain is a PICAXE-14A!

For more info & pictures, visit the *SERVO* Webstore. Tankbot and the Brain Alpha Kit can be purchased separately.

Combo Price \$ 138.95

Making Robots With The

Part 4 – Getting Feedback With Sensors

Arduino

By Gordon McComb

Robots need information about the world around them, or they just stumble about looking stupid. Just like us humans, a robot uses senses to know when it's run into something; when it's light or dark; when it's too hot or too cold; when it's about to fall over; or when it's found the way to the cheese at the center of a maze.

Senses require sensors. In the practice of robotics, the basic senses make do with the most basic of sensors: mechanical switches for detecting contact with objects, and photosensitive resistors and transistors for detecting the presence (or absence) of light. A robot can perform a remarkable amount of work with just the sense of touch and the gift of simple sight.

In this month's installment, you'll learn about interfacing switches and photosensors to the Arduino, along with how to use the information these sensors provide to interactively command a robot's motors. These are the fundamental building blocks of most any autonomous robot you build. Once you learn how to use these sensors to do your bidding, you can apply them in dozens of ways, for all kinds of robotic chores.

Arduino Robotics: What We've Covered So Far

This article builds upon previous installments in this series, which is all about the construction and use of the *ArdBot* (see **Figure 1**) — an inexpensive two-wheeled differentially-steered robot based on the popular Arduino Uno and compatible microcontrollers. If you'd like to follow along, be sure to check out the previous three episodes, so you're familiar with the plot and characters.

Part 1 introduced the *ArdBot* project, the Arduino, and basic programming fundamentals of this powerful controller.

Part 2 detailed the construction of the *ArdBot*, using common materials such as plastic or aircraft grade plywood.

Part 3 covered the Arduino in more depth, and examined the ins and outs of programming R/C servo motors with the Arduino.

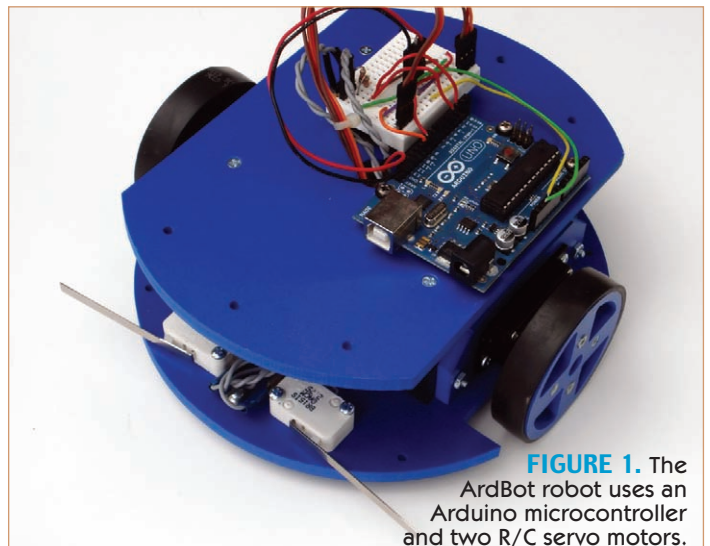


FIGURE 1. The *ArdBot* robot uses an Arduino microcontroller and two R/C servo motors.

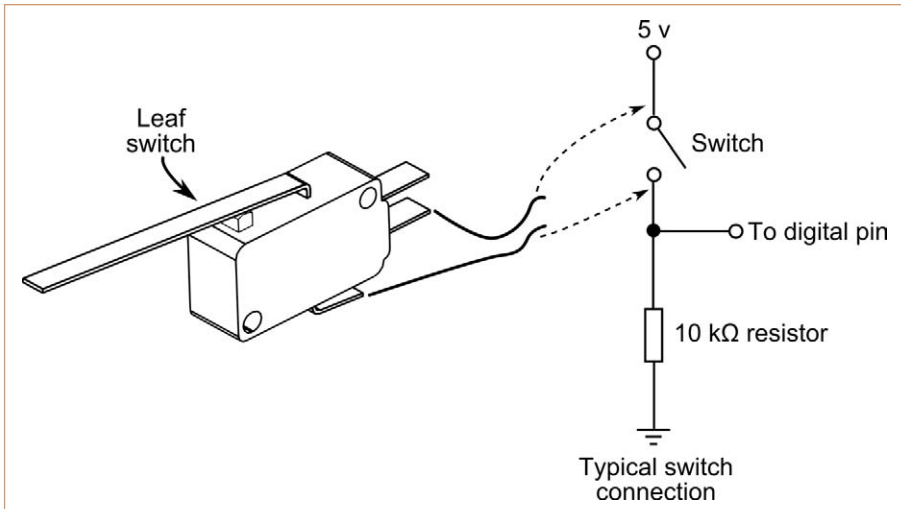


FIGURE 2. The leaf switch acts as a kind of cat's whiskers. It's connected to a digital I/O pin using a pull-down resistor.

Getting in Touch With Your Robot

Sensors — whether in humans or in robots — are designed to produce a reaction. What that reaction is depends on the nature of the sensation. Type and quantity matter. We interpret the feeling of a soft summer breeze as a good sensation. Increase the amount of air pressure to hurricane force and decrease the temperature to something below

freezing, and suddenly the same senses produce a highly negative reaction.

Touch — also called *tactile feedback* — is a primitive reactive sense. The robot determines its environment by making physical contact; this contact is registered through a variety of touch sensors. What happens when contact is made is entirely determined by the programming you apply within your robot.

Most often, a collision with an object is a cause for alarm. So, the reaction of the robot is to stop what it's doing, and back away from the condition. In other cases, contact can mean your robot has found its home base, or that it's located an enemy bot and is about to pound the living batteries out of it.

The lowly mechanical switch is the most common — and most simple — form of tactile (touch) feedback mechanism. Just about any momentary, spring-loaded switch will do. When the robot makes contact, the switch closes, completing a circuit.

I like to use *leaf* (or *lever*) switches (see **Figure 2**) because they function a lot like a cat's whiskers. These things are sometimes referred to as a microswitch — after a popular brand name — but I'll call them leaf switches to avoid confusion. Regardless of make or model, most are easy to mount, and come with plastic or metal strips of different lengths that enhance the sensitivity of the switch.

You can enlarge the contact area of the leaf by gluing or soldering bigger pieces of plastic or metal to it. For example, you can cut up some stiff music wire (available at hardware stores) or a cheap wire clothes hanger, and bend it to some fancy shape.

What's covered here applies to most any robot that uses the Arduino microcontroller, and that runs on two motors and rolls on wheels or tracks. You're free to adapt the techniques and programming code to whatever bots you're constructing. The ArdBot is an expandable platform, but it's also a concept that represents the typical desktop-size robot.

The subject of sensors is pretty involved. There's no way to cover all the interesting things in just one article.

So, next month you'll learn about other kinds of inexpensive sensors you can use with your ArdBot (or other robot).

Making Reusable Sensor Components

The more you experiment with robotics, the more you'll want to build a drawer full of reusable parts that easily plug into your projects. Sensors especially.

With just a bit of wire, some heat shrink tubing, and a length of snap-off male header pins you can build modular sensors that can be shared between projects. The pins easily plug into a solderless breadboard. **Figure A** shows a photoresistor attached to an eight inch length of wire which is terminated into a three-prong male header (only two pins are used; the third is cut off).

First, start by cutting some 22 or 24 gauge insulated stranded conductor wire to the desired length. Don't be stingy with the wire, but don't make it so long the extra gets in the way. Give yourself an additional inch or two so you can twist the leads together to make a nice pigtail.

Strip about 1/4" insulation off both ends, and use your soldering pencil to pre-tin the wire. Do the same for the leads on the photocell and the header pins. Exercise care when soldering to the photocell leads, as excessive heat can damage the component. After tinning is complete, carefully tack-solder the wires to the leads or pins.

I like to use heat shrink tubing to finish off the soldered ends. The tubing



FIGURE A. Use male header pins to prepare connectors for easy interchange with your various projects. The connectors plug into a solderless breadboard.

makes for a more professional look, plus it helps prevent short circuits. When applied properly, it acts as a strain relief to help keep the wires from pulling apart from their joints. Buy a small assorted package of tubing, and use the smallest diameter for the best fit.

Header pins have a 0.100" spacing which is a fairly tight space for all but the most seasoned solder pros. So, you'll probably want to snap off a set of three or four pins, and remove the center pins to make extra room for your solder joints.

FIGURE 3. A pair of leaf switches on the ArdBot. You can attach things to the leaf of each switch to enlarge its contact area.

Solder the end(s) to the leaf. Or, you can use thin pieces of wood, plastic, or metal. Just be sure the weight of the extension doesn't accidentally activate the switch. You don't want false alarms.

The switch may be directly connected to a motor or, more commonly, it may be connected to a microcontroller. A typical wiring diagram for the switch is shown in **Figure 2**. The 10 k Ω pull-down resistor is there to provide a consistent digital LOW (0 volts) output for the switch when there is no contact. When contact is made, the switch closes, and the output of the switch goes HIGH — usually five volts, as shown here.

Using Leaf Switches as Bumpers

Two standard leaf switches mounted to the front of your ArdBot let it detect when it's hit something. With the switches situated to the sides, your bot can determine if the object is on the left or on the right, and then steer around it.

Figure 3 shows a pair of leaf switches mounted like bumpers to the front of the ArdBot. Switches like these are available at many online electronics outlets, and are common as surplus. I bought these at All Electronics (a *SERVO Magazine* advertiser) for \$1.60 a pop.

I haven't augmented the switches with a larger contact area, as I'm more interested in demonstrating the concepts involved. Use your creativity in enhancing the switches to provide the level of sense detection you want. For example, right off you can see that the robot is "blind" to small objects directly between the switches. You can deal with this either by enlarging the contact area, or (my choice) using another form of "sense" to avoid collision in the first place.

To mount each switch, find two suitable holes in the base of your robot, or drill new ones. Most leaf switches have three connections: common, normally open (NO), and normally closed (NC). Wire the *common* and *NO* connections. If space is tight, break off the NC connection to make room.

Figure 4 shows the diagram for connecting the two switches to digital pins D2 and D3 of the Arduino. **Figure 5** shows the same circuit, but in breadboard view. Use the upper half of the ArdBot's 170 tie point solderless breadboard. The bottom half is already in use by the servo wiring for the ArdBot (see Part 2 of this series).

On my prototype, I made connectors for the switches by soldering the two wires to pins of a breakaway male header. With these, you break off the number of pins you

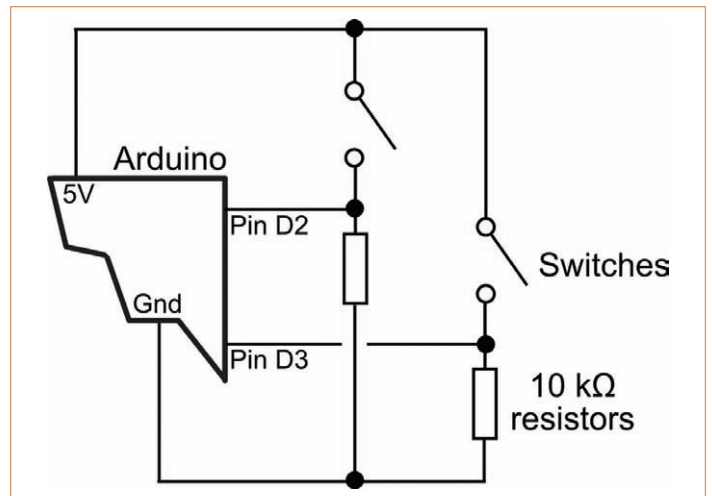
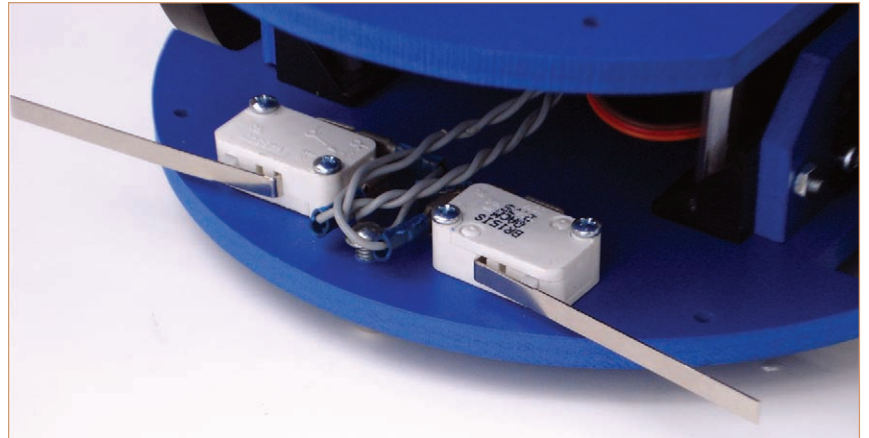


FIGURE 4. Schematic view of connecting two bumper switches to the Arduino microcontroller.

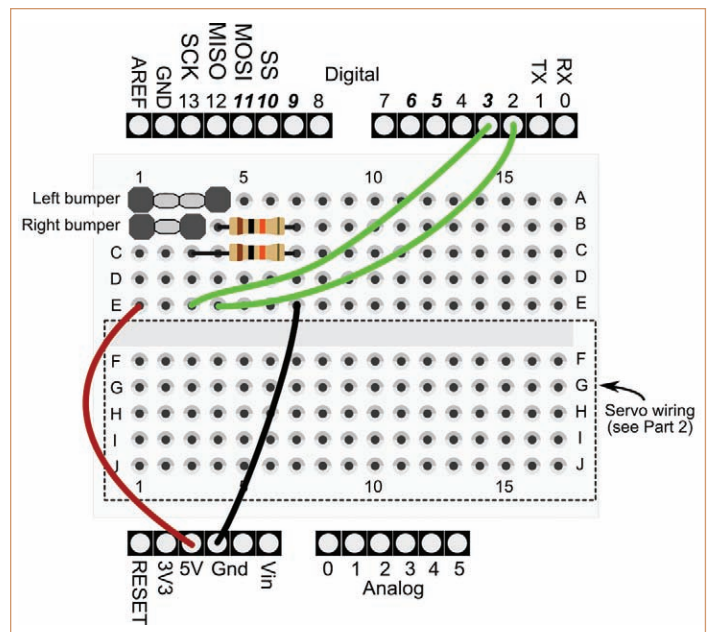


FIGURE 5. Breadboard view of connecting the bumper switches. Note that the bottom half of the solderless breadboard is already in use, wired for the two servo motors. (See Part 2 of this series for details.)


```

/*
ArdBot bumper switch demo
Requires Arduino IDE version 0017
or later (0019 or later preferred)
*/

#include <Servo.h>

const int ledPin = 13; // Built-in LED
const int bumpLeft = 2; // Left bumper pin 2
const int bumpRight = 3; // Left bumper pin 3
int pbLeft = 0; // Var for left bump
int pbRight = 0; // Var for left bump
Servo servoLeft; // Define left servo
Servo servoRight; // Define right servo

void setup() {
  servoLeft.attach(10); // Left servo pin D10
  servoRight.attach(9); // Right servo pin D9
  // Set pin modes
  pinMode(bumpLeft, INPUT);
  pinMode(bumpRight, INPUT);
  pinMode(ledPin, OUTPUT);
}

void loop() {
  forward(); // Start forward
  // Test bumper switches
  pbLeft = digitalRead(bumpLeft);
  pbRight = digitalRead(bumpRight);

  // Show LED indicator
  showLED();

  // If left bumper hit
  if (pbLeft == HIGH) {
    reverse();
    delay(500);
    turnRight();
    delay(1500);
  }

  // If right bumper hit
  if (pbRight == HIGH) {
    reverse();
    delay(500);
    turnLeft();
    delay(1500);
  }
}

// Motion routines
void forward() {
  servoLeft.write(180);
  servoRight.write(0);
}

void reverse() {
  servoLeft.write(0);
  servoRight.write(180);
}

void turnRight() {
  servoLeft.write(180);
  servoRight.write(180);
}

void turnLeft() {
  servoLeft.write(0);
  servoRight.write(0);
}

void stopRobot() {
  servoLeft.write(90);
  servoRight.write(90);
}

void showLED() {
  // Show LED if a bumper is hit
  if (pbRight == HIGH || pbLeft == HIGH) {
    // Turn LED on
    digitalWrite(ledPin, HIGH);
  }
  else {
    // Turn LED off
    digitalWrite(ledPin, LOW);
  }
}

```

Listing 1 – bumper.pde.

want to use. I cut one connector to three pins wide, removing the middle pin; I soldered the wires from the switch to the outer two pins. For the other connector, I cut it to four pins wide, removing the middle two pins. You can see in **Figure 5** how the two connectors plug into the breadboard.

Important! Make sure all the wires and other components are firmly seated into their breadboard tie-point sockets. Loose connections are the second most common cause of problems when using a solderless breadboard — the most common is plugging the wires into the wrong tie points!

Listing 1 shows the demo program *bumper.pde*. The ArdBot sets off going forward until one of its front bumper switches makes contact with an object. The moment the switch closes, the robot quickly reverses direction, then turns in the opposite direction of the obstacle. Time delays are specified in milliseconds. The robot backs up for 500 milliseconds (half a second). It then turns — actually spins — to the right or left for 1,500 milliseconds (1.5 seconds).

You can experiment with other delay settings, depending on how fast your robot travels. With faster servo motors, you can use a shorter delay. The idea is to spin the robot about one-quarter to one-half turn, so it moves away from the obstacle.

Note that “left” and “right” (and “front” and “back”) are somewhat objective in a robot like the ArdBot. Either end can be the front, so left and right is relative. In my prototype, I put the two leaf switches on the end that had more mounting space available. That end became the “front.” The coding in *bumper.pde* reflects this design choice. If your robot seems to behave opposite to what it should, swap the values in the motion routines (forward, reverse, etc.). See Part 3 of this series for more details on what the servo commands do, and how they work.

Understanding the bumper.pde Sketch

As with all Arduino sketches, *bumper.pde* has three principle parts: declaration, *setup()* function, and *loop()* function.

The *declaration* area at the top of the sketch sets up the variables used throughout the program. It also prepares two objects of the servo class. As you read in Part 3 of *Making Robots with the Arduino*, the servo class is provided as a library that comes with the Arduino programming tools. You use it to control one or more R/C servos. The declaration also defines the two leaf switches as connected to digital pins D2 and D3, and that we’ll be using the Arduino’s built-in LED (internally connected to pin D13) as a visual indicator.

In the *setup()* function, the servos are defined as connected to digital pins D9 and D10. The pins used for the LED and two switches are set as outputs and input, respectively.

The main body of the sketch is the *loop()* function

which repeats indefinitely. It begins by activating the two servos to move the robot forward. The sketch then uses the *digitalRead* statement to store the current state of the two switches. The instantaneous value of the switches is kept in a pair of variables (*pbLeft* and *pbRight* — the *pb* for pushbutton). These variables are used elsewhere.

Of main interest in the *loop()* function are the two *if* statements. Here's the one that tests the left leaf switch:

```
if (pbLeft == HIGH)
  reverse();
  delay(500);
  turnRight();
  delay(1500);
}
```

pbLeft == HIGH checks to see if the contents of the *pbLeft* variable (set earlier based on the state of the left leaf switch) is HIGH. If it is, then the left switch is closed, and the robot has made contact with something. If it's LOW, then the switch is open, and the robot continues on its way.

Bumper.pde also includes a number of user-defined functions. Most — like *forward()* and *reverse()* — relate to driving the servo motors. Another function, *showLED()*, toggles the LED on pin D13 of the Arduino on or off, depending on whether a switch is closed. Use this as a visual indicator that the programming code is working as it should.

Switch Triggers Using Polling or Interrupts

The programming in *bumper.pde* relies on what's known as *polling*: The sketch repeatedly checks the status of the two switches. If a switch is closed, its value goes from LOW to HIGH; when HIGH, the robot is commanded to steer to a new heading. The switches are checked — polled — many times each second.

Polling is an acceptable method when the sketch is relatively simple, and the demands on the Arduino are light. For code that is more processing intensive, there is a remote chance the controller will miss when a leaf switch has closed. It'll be busy doing something else in between polls and be unaware anything has happened.

In truth, you can have a fairly involved sketch and it will still detect 99 percent of all switch closures. The reason: The switch will likely be closed for what are very long periods of time to a microcontroller. For a microcontroller running at 16 MHz, even a brief 100 millisecond (one-tenth second) contact is like a lifetime, and so in all likelihood the switch closure will be registered.

Still, if you absolutely must ensure that even the most fleeting contact is registered, you might want to consider using *hardware interrupts* rather than polling. With an interrupt, special code is run if — and only when — a specific external event occurs. Because the main program

```
/*
ArdBot interrupt bumper demo
Requires Arduino IDE version 0017
or later (0019 or later preferred)
*/

#include <Servo.h>

const int ledPin = 13;
const int bumpLeft = 2;
const int bumpRight = 3;
int pbLeft = 0;
int pbRight = 0;
Servo servoLeft;
Servo servoRight;

void setup() {
  servoLeft.attach(10);
  servoRight.attach(9);
  // Set pin modes
  pinMode(bumpLeft, INPUT);
  pinMode(bumpRight, INPUT);
  pinMode(ledPin, OUTPUT);

  // Set up interrupts
  attachInterrupt(0, hitLeft, RISING);
  attachInterrupt(1, hitRight, RISING);
}

void loop() {
  forward(); // Start forward
  showLED(); // Show LED indicator

  // If left bumper hit
  if (pbLeft == HIGH) {
    reverse();
    delay(500);
    turnRight();
    delay(1500);
    pbLeft = LOW;
  }

  // If right bumper hit
  if (pbRight == HIGH) {
    reverse();
    delay(500);
    turnLeft();
    delay(1500);
    pbRight = LOW;
  }

  // Motion routines
  void forward() {
    servoLeft.write(180);
    servoRight.write(0);
  }

  void reverse() {
    servoLeft.write(0);
    servoRight.write(180);
  }

  void turnRight() {
    servoLeft.write(180);
    servoRight.write(180);
  }

  void turnLeft() {
    servoLeft.write(0);
    servoRight.write(0);
  }

  void stopRobot() {
    servoLeft.write(90);
    servoRight.write(90);
  }

  void showLED() {
    // Show LED if a bumper is hit
    if (pbRight == HIGH || pbLeft == HIGH) {
      digitalWrite(ledPin, HIGH);
    }
    else {
      digitalWrite(ledPin, LOW);
    }
  }

  // Interrupt handlers
  void hitLeft() {
    pbLeft = HIGH;
  }

  void hitRight() {
    pbRight = HIGH;
  }
}
```

Listing 2 – interrupt.pde.

`loop()` doesn't have to continually check the state of the pins, it frees up the controller to do other things. Reaction time of an interrupt is measured in microsecond timing, even if the Arduino is busy doing something else. (Actually, this is not always true, depending on how other hardware on the controller is being used. But any additional delay is usually minimal.)

The Arduino Uno supports two hardware interrupts (the Arduino Mega supports six) internally connected within the Arduino to digital pins D2 and D3. These are the pins that the leaf switches are already connected to, so the only change needed is in the software.

See **Listing 2** for *interrupt.pde*. Here, the *bumper.pde* sketch has been revised to "listen" to a state change on both of the hardware interrupts with the statements:

```
attachInterrupt(0, hitLeft, RISING);
attachInterrupt(1, hitRight, RISING);
```

Note that the interrupts are referred to as 0 and 1. These correspond to pins D2 and D3, respectively. The labels *hitLeft* and *hitRight* are the functions that are called when the interrupt is triggered. Finally, *RISING* is a built-in constant that tells the Arduino to trigger the interrupt on a LOW-to-HIGH signal transition. This type of transition occurs when the switch closes.

Both *hitLeft* and *hitRight* set their corresponding "pb" variable to HIGH. The program then immediately exits the interrupt handler. The next time the Arduino repeats its *loop()*, it notices that a pushbutton is HIGH and performs the needed obstacle avoidance maneuver. (Note also that the pushbutton value is manually set back to LOW, in anticipation of the next bump.)

You might be asking why the code to control the servos isn't in the interrupt handlers. The reason is this: The

delay statement — which is used to steer the robot around an obstacle — is disabled while in an interrupt. In any case, it's usually best not to place time-intensive functionality within interrupt handlers.

To Let Bounce or Debounce?

In a perfect world, mechanical switches would produce clean, reliable digital signals for our microcontrollers. We don't live in a perfect world; instead, we must suffer something called *switch bounce*. Instead of a nice LOW-to-HIGH digital pulse when a switch closes, what we get are five, 10, maybe even dozens of irregular glitches, all caused as the metal contacts in the switch settle into position. All this happens very quickly; usually in just a few milliseconds.

For some applications, it's absolutely necessary to *debounce* the output of a switch. In debouncing, all the glitches are removed, providing the microcontroller with that single sweet pulse we want. You can debounce with some extra hardware: a capacitor and resistor create an RC network that acts to delay the rise and fall of the switch signal — that effectively removes the glitches. You can also do it in software, typically using delays so that the microcontroller ignores all but the first signal transition.

Both the *bumper.pde* and *interrupt.pde* examples don't directly use switch debounce. Software delays are already built into the code, plus R/C servos are slow creatures and don't react fast enough for bounce to be a problem. Servos are commanded in 20 millisecond "frames" — that is, their operation is updated once every 20 milliseconds. Turns out that's about the worst-case duration of bounce glitches from most switches. So, even with a switch bouncing along merrily, it has little or no effect on the operation of the servos.

Should you need to debounce your switch inputs, there's a separate class library you can download and use with the Arduino. The library is called *Bounce*, and it's available from the main Arduino language reference pages. There's also a *Debounce* code example that comes with the Arduino programming IDE.

Mounting Alternatives, More Switches

A few quick notes before moving on. So far, I've talked about the two switches in the front of the robot, situated right and left. Feel free to put switches anywhere you want. You might instead have a front and back switch, or a bunch of switches all around the periphery of the robot.

If you use more than two switches, you'll have to rely on polling, as there are only two pins that support hardware interrupts (when using the Arduino Uno). If you use more than four or five switches, you may want to use a

Sources

If you'd like to build the ArdBot, be sure to start with the November '10 issue of *SERVO Magazine* for Part 1 of this series. Also check out the following sources for parts:

Example code and more:

Arduino
www.arduino.cc

Fritzing
www.fritzing.org

Prefabricated ArdBot body pieces with all construction hardware:

Budget Robotics
www.budgetrobotics.com

Partial list of Arduino resellers:

AdaFruit
www.adafruit.com

HVW Tech
www.hvwtech.com

Jameco
www.jameco.com

Pololu
www.pololu.com

Robotshop
www.robotshop.com

Solarbotics
www.solarbotics.com

Sparkfun
www.sparkfun.com

FIGURE 6. By connecting another resistor in series with a photoresistor, the output is converted to a varying voltage. In this particular arrangement, the voltage increases under stronger light.

parallel-to-serial (*PISO*) shift register chip, such as the 74HC165 or CD4021. These are integrated circuits that take eight parallel inputs and provide a serial data output that can be read by the Arduino. Assuming eight switches, the PISO reduces the number of required I/O pins from eight to three. There's a code example for how to do this at arduino.cc/en/Tutorial/ShiftIn.

Let There Be Light (and let your ArdBot see it!)

Next to tactile feedback, reacting to light is the most common robotic sense. In lieu of actual vision, the robot uses electronic components such as photoresistors and phototransistors that are sensitive to light. Your bot may react to the simple absence or presence of light, or it may be able to measure the brightness, color, or other qualitative aspect of the light.

Photoresistors — also called photocells, light dependent resistors, or CdS (for cadmium sulfide) cells — are perhaps the easiest to use as simple light sensors. The photocell is a resistor whose value changes depending on the amount of light that strikes its sensing surface. In darkness, the photocell has a high resistance, typically in the neighborhood of 100 k Ω to over one megohm, depending on the component. The resistance falls as more light strikes the cell. In high brightness, the resistance may be as low as 1 k Ω to 10 k Ω .

The exact dark/light resistance values differ depending on the component, and even among photocells of the same make and model. Typical value tolerance is 10 to 20 percent. You can purchase photocells new, but they're common finds in the surplus market. Get a variety pack, and use your multimeter to "grade" each one. Cracks and other injury spell doom to a photocell; air and moisture degrade the sensing surface, rendering it useless. Toss any that don't react properly to a nearby desk lamp.

Being a resistor, you can convert the output of a photocell to a varying voltage merely by connecting another resistor to it in series, as shown in **Figure 6**. The value of the series resistor depends on the dark/light resistance range of the photocell, and how you want to use it. The cells I used had a dark resistance of about 40 k Ω and a light resistance of 30 ohms. In average room brightness, the cells had 10 k Ω resistors, so I selected a 10 k Ω series resistor.

The voltage at the point between the photocell and series resistor is a ratio of the two resistance values. With the two resistances equal, the divided voltage between them is one-half of the supply voltage; in the case of five volts in average room light, the output voltage is 2.5 volts.

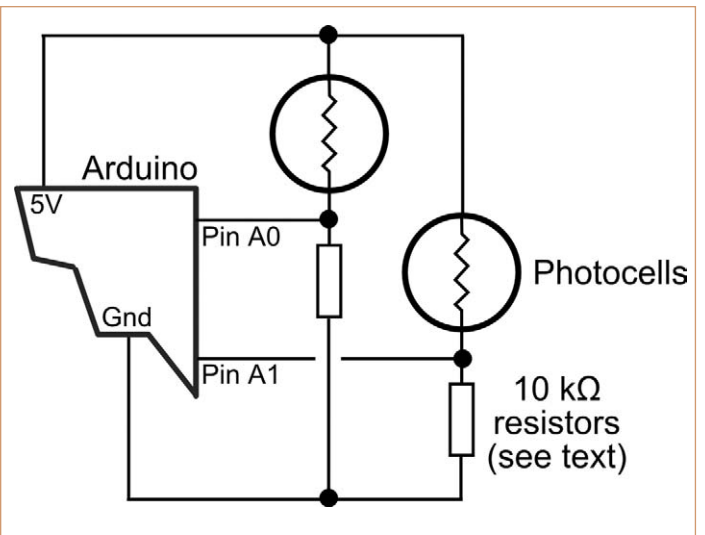
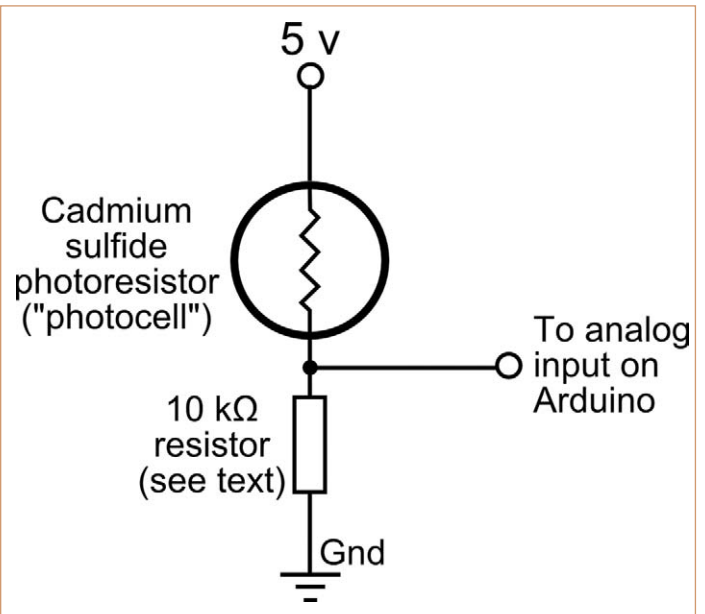


FIGURE 7. Schematic view of connecting two photocells to the Arduino microcontroller.

Listing 3 – simplecds.pde.

```
/*
  ArdBot CdS cell demo
*/

int cds = 0;

void setup() {
  Serial.begin(9600);
}

void loop() {
  // Read analog pin A0 and display value
  // on Serial Monitor window
  cds = analogRead(A0);
  Serial.println(cds, DEC);
  delay (200);
}
```

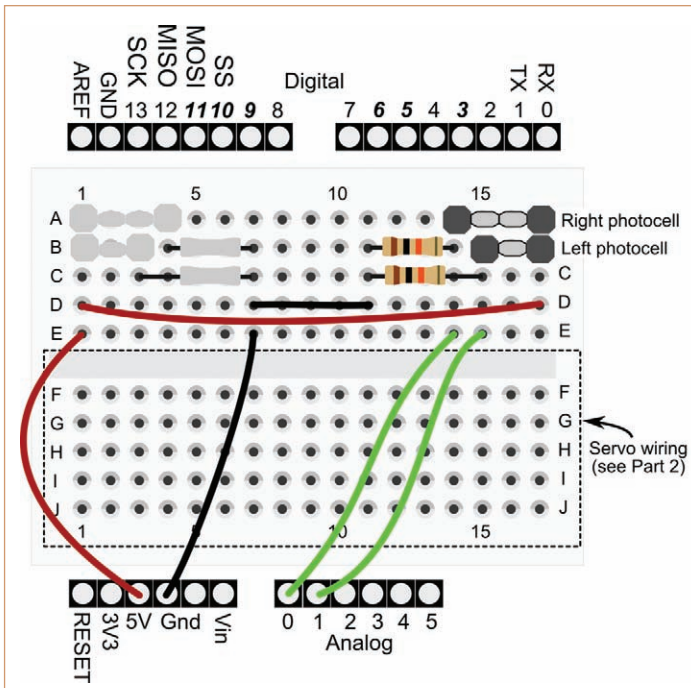



FIGURE 8. Breadboard view of connecting the photocells.

The voltage decreases in darkness and increases as more light strikes the photocell.

You will need to experiment with the series resistor to determine its best value, based on the specific photocells you use. You might want to try a 50 k Ω or 100 k Ω potentiometer in place of a fixed resistor, allowing you to fine-tune the series resistance as needed.

Listing 3 shows *simplecds.pde*, a basic sketch that tests the operation of the photocell. Wire the photocell as shown in **Figure 6** and connect the output to analog pin

A0 of your Arduino. Compile and upload the sketch, then open the serial monitor window. You'll see a series of numbers; they correspond to the output voltage of the sensor converted to a 10-bit (0 to 1023) numeric value. You should get a low number when all light to the photocell is blocked, and a higher number under full illumination.

Steering Your Robot With a Flashlight

By using two photocells mounted on each side of your ArdBot, you can literally steer it by flashlight. Under just room light, the robot is set to stop, waiting for your command. Aim the flashlight so that light falls more or less equally on both photocells, and the robot will move forward. When the light levels aren't equal, the robot will turn toward the photocell that has more light falling on it.

Refer to **Figure 7** for a schematic of the two-photocell setup. **Figure 8** shows the same circuit but in breadboard view. For my prototype, I made small mounts for the photocells using scrap PVC plastic, then attached the mounts to the top deck of the ArdBot with metal brackets. The photocells I used measured 0.29" x 0.25" (elliptical shape). I drilled holes just slightly smaller, then used a rat-tail file to enlarge the holes so that the cell just fit inside. On my prototype, the cells are held in just by friction, but on yours you can use hot-melt glue or other adhesive that when set leaves no moisture for a possible short circuit.

(Bear in mind light can strike the photosensitive surface of the cell from the rear. You may want to add a layer or two of black tape to prevent light spoilage.)

Figure 9 shows my ArdBot with the two photocell "eyes" attached to the front. I've bent the brackets back a bit so that the cells point slightly upward.

Refer to **Listing 4** for *lightsteer.pde*. It uses the current values of the photocells to make quick steering adjustments to the left or to the right. In the declarations area, the code:

```
const int ambient = 600;
const int threshold = 800;
```

sets two comparison values used elsewhere in the sketch. *You will need to experiment with these values depending on the room environment and photocell characteristics!* These values worked for me; you can start with them, but expect to try other values as you fine-tune the performance of the steering.

The *ambient* value sets the upper level of just the ambient (natural) light in the

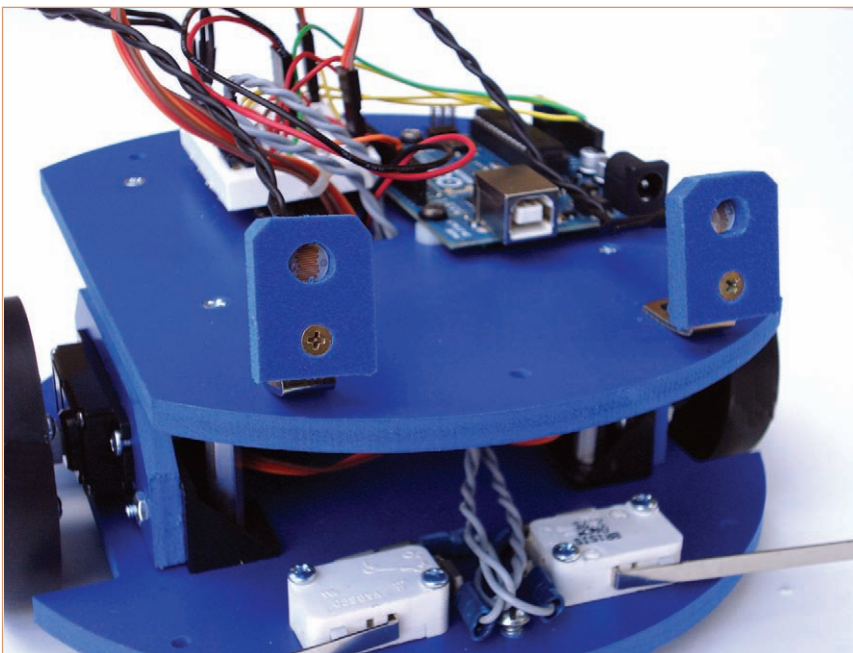


FIGURE 9. Mount the photocells on the top of the ArdBot — toward the left and right sides — to make eyes for following the flashlight beam.

room. This is the amount of light that hits the photocells under normal lighting conditions. For me, the ambient light value was about 520 to 530, so I made it a little higher (600) for extra headroom.

The *threshold* value sets the lower level of the light beamed from your flashlight. I set the value at 800 based on using a nine-LED flashlight (with fresh batteries), 1-2 feet away from the robot. For best results, use a bright flashlight near the robot — the farther away you get, the less light that falls on the photocells.

More Light Tricks

Just by switching around some of the code you can have your robot run away from you rather than try to follow you. Or, by placing colored gel filters over the photocells and using a color LED flashlight (blue and green are popular), your robot can more readily discriminate between light to follow and light to ignore.

You might also add one or two “room light” sensors that aren’t in the direct line-of-sight of the flashlight beam. These could be used to set the ambient light level of the room.

Small lenses over the photocells help to focus the flashlight beam, improving steering performance and helping the bot reject any light not directly to the front.

These are just some of the things you can do to give your ArdBot (or other Arduino-powered bot) the gift of sight. Feel free to experiment. Photocells and other light sensitive components are inexpensive, and changing the code in an Arduino sketch is absolutely free.

Coming Up ...

In our next installment, you’ll discover even more senses you can provide, including ultrasonic sound for measuring distances to objects, and infrared light to determine the proximity of nasty things that are in the way.

You’ll also read about ways for your robot to scan the room to soak up its environment, rather than just see life through a narrow tunnel in front of it. Exciting stuff (at least for your ArdBot), so don’t miss it! **SV**

Listing 4 – lightsteer.pde.

```
/*
ArdBot steering by light demo
Requires Arduino IDE version 0017
or later (0019 or later preferred)
*/

#include <Servo.h>

// CdS cell reference values
// (you need to experiment)
const int ambient = 600;
const int threshold = 800;

int lightLeft = 0;
int lightRight = 0;

Servo servoLeft;
Servo servoRight;

void setup() {
  servoLeft.attach(10);
  servoRight.attach(9);
}

void loop() {
  // Read light sensors connected to
  // analog pins A0 and A1
  lightLeft = analogRead(A1);
  lightRight = analogRead(A0);

  // Stop robot if below ambient
  if (lightRight < ambient || lightLeft < ambient) {
    stopRobot();
  } else {
    forward();
    // Steer to right if right CdS below threshold
    if (lightRight < threshold) {
      turnLeft();
      delay (250);
    }
    // Steer to left if left CdS below threshold
    forward();
    if (lightLeft < threshold) {
      turnRight();
      delay (250);
    }
  }
}

// Motion routines
void forward() {
  servoLeft.write(180);
  servoRight.write(0);
}

void reverse() {
  servoLeft.write(0);
  servoRight.write(180);
}

void turnRight() {
  servoLeft.write(180);
  servoRight.write(180);
}

void turnLeft() {
  servoLeft.write(0);
  servoRight.write(0);
}

void stopRobot() {
  servoLeft.write(90);
  servoRight.write(90);
}
```

Gordon McComb can be reached at
arduino@robotoid.com.



Then and NOW

AMAZING ROBOTS ARISE FROM JUNK

b y T o m C a r r o l l

There has always been one topic in my column over the years that seems to have generated the most reader response, and that is the 'early' hobbyist or experimenter-built robots. With all of the robot kits and parts available to robot hobbyists today, it always seems to be amazing just how people built such creative robots with none of the technology that we take for granted today. Like the mythical Phoenix of lore, these early robotic creations have arisen from the most basic of materials to perform incredible feats.

There have been some interesting threads on the Seattle Robotics Society's (SRS) and the Portland Area Robotics Society's (PARTS) list servers about various old robots, in particular, the Szegedi Katicabogár or Ladybird robot. This is one of many hundreds of innovative robots built back in the 1950s and '60s that exhibited some fairly sophisticated behaviors — especially considering the crudeness and lack of modern electronics in those days. I'd like to examine this one and others to find out just what made these early machines so unique.

Robot Parts Evolve Over 50 Years

In those days, robot experimenters had to use dry cell 'D' batteries or maybe some small lead-acid batteries as power sources. There were very few DC motors available and windshield wiper motors from cars were about the only gearmotors available to the robot hobbyist. Although there were a few electric seat and window motors from high end cars. Cadmium sulfide cells were just becoming available for robot eyes and silicon diodes were replacing selenium rectifiers to convert AC to DC. Junked juke boxes had some great robot parts and those lucky enough to live close to surplus businesses had a treasure trove of old war surplus aircraft DC gearmotors and linear actuators nearby. Plywood and fiberglass sheets were the construction materials of choice, though galvanized steel ducting was handy for robot bodies. Christmas tree lights (the series types) were the only low voltage bulbs available

besides car bulbs.

As a kid, I would spend hours in the local library in my small North Carolina town of Mount Olive, trying to find information about *any* type of robot and how they were built. The back pages of *Popular Mechanics* and *Popular Science* had tiny ads where people would have to send a post card or letter in order to receive a catalog, in hopes that it would feature things useful for robot building. If I was lucky enough to get to the big city of Raleigh, I would find myself lost in a building of rusting and dusty surplus junk — a goldmine to me. To order something, I would send a letter or order form filled out, along with a check to someplace in Timbuktu. Then, weeks later, the item would arrive. Sometimes it was exactly what I wanted, but sometimes it was nothing like what was described in the catalog.

A Leap to the Present Time

Fifty years later, I can Bing or Google any subject I want, and up pops tens of thousands of sites. A few deft key strokes on my computer and I am winding my way through all sorts of robot goodies. Another few key strokes and fabulous products are being shipped my way. We are so lucky today to be able to use a microcontroller that costs about a dollar that can replace dozens of relays and a handful of vacuum tubes. (Remember those?) The best part is that now there are many robot-specific sites and dealers available.

FIGURE 1. Dr. Walter working on a later version of Elmer.

High-power density and lightweight LiPoly batteries, efficient H-bridge power controllers for our robot's motors, and a vast number of sophisticated sensors are available to us today that would have amazed scientists 50 years ago. GPS, RF links, LEDs, finger-sized HD color cameras, laser navigation, and that one thing that made small robots so easy to build — the model aircraft servo — all added to the mother lode that enhanced experimenter's robot building. Add to that the many microcontrollers and inexpensive microprocessor-based computers and one can build some remarkable robotic devices.

Homebrew Robotics

There is one main attribute that sets experimental and hobbyist robotics apart from the typical computer user or even hacker activities and that is the homebrew aspect. Robot hobbyists like to *build* things with their hands, not just plug some module into their computer and write a hundred lines of code for it. Yes, code is important for any smart robot but interfacing different motors, sensors and feedback devices with the 'brain' is what makes robotics so interesting. Designing appendages and joints, accurate placement of the sensors, weight and balance, and power requirements all come into play to make robot building so rewarding.

Early Robot Experimenters

Going back six decades, in 1948 to 1949, Englishman William Grey Walter built two phototropic (light seeking) robot turtles. The most famous one he called Elsie as it had a clear shell, unlike his earlier Elmer that used strips of folded tin to replicate a shell. **Figure 1** shows Walter's lab and an improved Elmer. It is quite interesting to read what a newspaper said about the earlier Elmer: "LONDON. Feb. 25, 1948. The *Daily Express* reported today that Dr. W. Grey Walter, 38 year old brain scientist, has built a robot tortoise so human that it likes company, recognizes voices, and comes to heel when called. The paper said that the tortoise avoids cold or damp weather, great heat or bright lights, likes women but dislikes men. "It can be temperamental and will be neurotic and sulky for days if teased or given too many contradictory instructions." Dr. Walter, at the Burden Neurological Institute in Bristol, England, built the tortoise with ordinary radio valves (tubes), switch relays, miniature microphones for registering sound, and photoelectric cells for recognizing color and shape." (That sounds even better than Sony's Aibo!) This type of hype was typical for any new fangled gadget that made the headlines in those days. The news media made it sound so much better than reality.

As a neurophysicist, Walter was interested in conditioned responses and needed a 'tool' to exhibit these

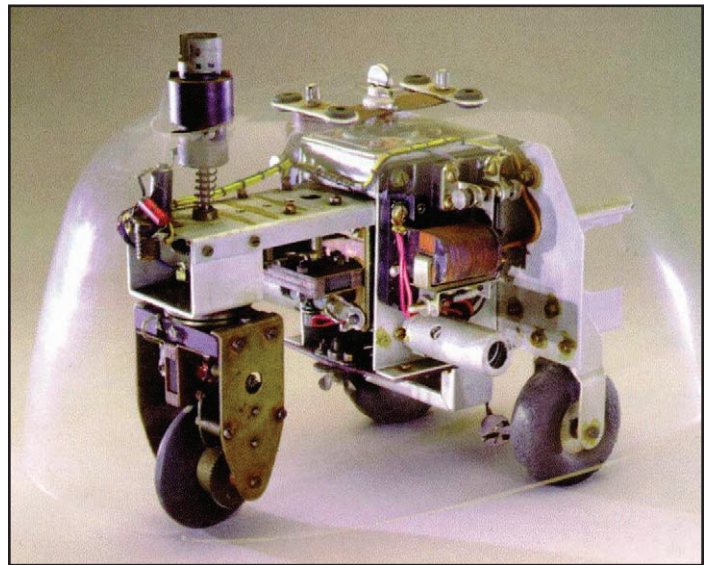


FIGURE 2. The Bristol Tortoise.

responses while working at the Burden Neurological Institute. Walter's original robots — built with the help of his wife, Vivian — were dismantled and the various parts were used in improved models. Only photographs remain, though personnel of the BNI built several replicas. **Figure 2** shows of an accurate replica called the Bristol Tortoise, though the motors are of a more modern variety. His machines were considered the first true robots and many people copied



FIGURE 3. Daniel Muszka's Katicabogár.

and improved upon his work. (I covered Grey Walter's turtles in more detail in my October '05 column.)

The Elektronikus Katicabogár or Electronic Ladybird

One of the most noteworthy experimental robots to follow the pattern of Walter's work was the Electronic Ladybird or, as others called it, a ladybug. It was built by Dániel Muszka and László Kalmár at Szeged University, Hungary. A replica of the original is shown in **Figure 3**. Note the differences between Walter's robot and the newer ladybird robot — especially the complexity shown in **Figure 4**. The cybernetic animal is now on display at the Informatika Történeti Múzeum Alapítvány at Szeged, Hungary. The robot is 60 cm or almost 24 inches long, 16 inches wide, and 10 inches high, and is painted with red

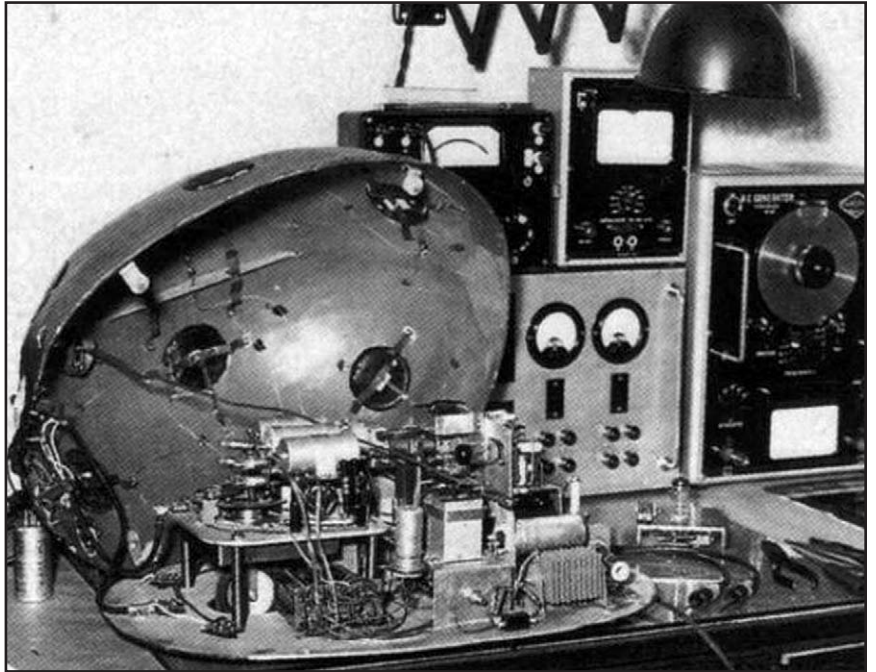


FIGURE 4. Ladybird's insides (in 1957).

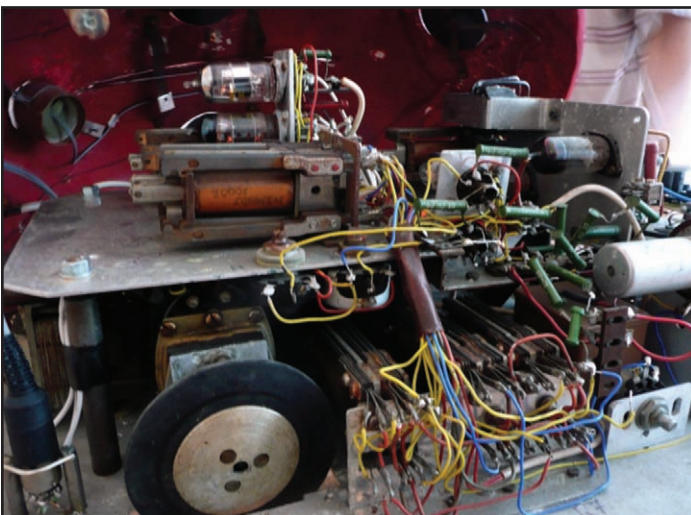
and black dots like a ladybird beetle. The seven black dots on its shell are touch-sensitive switches. The robot is powered externally by a 240 VAC power cord trailing out the back as the tubes in later models drew too much power to use a battery-powered system like the original model.

Figure 5 shows the tube's filaments brightly glowing.

The ladybird robot was built to model conditioned reflexes, much the same as Walter's turtle almost a decade earlier. The robot was also phototropic and could sense light from a flashlight, sounds from a whistle, and touching on the various black dot switches. The 'eyes' were just red lights but three light sensors were at the front of the robot: one centered between the eyes, and two just outside the black face. Depending on which light sensor was illuminated by an external source determined whether the robot continued straight ahead or turned to the right or left. A microphone mounted on the shell sensed a whistle to command starting and stopping. As with many of the older one-of-a-kind classic robots, the electronic ladybird suffered some mishandling by a museum and its original papier mâché shell was damaged beyond repair, so a new fiberglass shell was made. The internal motors and circuitry are the original components, however. In 2004, a modern reproduction was made, more or less the same externally, but nothing like the original on the inside. Though complex-appearing due to the large components, the robot was basically a simple electronic device. There are only seven vacuum tubes (compared to only two in Walter's turtle), three crystal diodes, three photoelectric cells, one microphone, and just two motors in the robot.

The primary operational stimulus/response is to locate a light source and home in on it. As it is in its random search pattern, the Coccinella (as it is sometimes called)

FIGURE 5. Power-hungry tubes draw power.



determines if either of the two side light-sensing photoelectric cells see a light from a hand held flashlight. At this point, the random pattern stops when the driven wheel on the side of the illuminated photocell stops. This causes the other wheel to curve the 'machina docilis' (as it is also called) towards the axis of the light beam. When the central photoelectric cell acquires the light beam, the forward motion is stopped and the ladybird is centered on the flashlight's beam.

Pressing some or all of the dots on the robot's shell causes the robot to either start moving again, to stop, or to even emit a soft murmuring sound. If the back is stroked again (I assume that there is a switch located up there), another response is starting movement and ceasing the sound. A whistle with a specific tone causes the eyes to light up or turn off. Muszka and Kalmár also used a unique type of memory — a capacitor — to store a conditional reflex or stimuli response. When the capacitor is fully charged, it represents a certain state, and when it slowly discharges on its own (as capacitors do), another state is reached. That might trigger a new searching movement or cause it to go to sleep. This robot in its shiny new shell continues to entertain and educate people at the Informatika Történeti Múzeum Alapítvány.

Robot Dogs

No discussion on animal-behavior robots would be complete without mentioning robot dogs. Sony's Aibo was all the rage 10 years ago, and possessed some amazing capabilities, even considering today's advanced electronics and sensors. Over half of all animal-like robots have been in the form of dogs. John Hammond, Jr. and Benjamin Miessner developed the Electric Dog in 1912 — the first

selenium photocell phototropic animal that was a test bed for further development of war-time torpedoes and such. Walter's turtles and the Hungarian ladybird that came later were research vehicles to prove certain neurological phenomena not for weapon test beds. The Philips Dog from 1958 was a unique electronic marvel for its period in history and was used as a programming test bed.

The Philips Dog

Built under the auspices of the Dutch company, Philips, Anne Hendrik Bruinsma's creation shown in **Figure 6** was chock full of electronics. These photos were taken from her book entitled *Practical Robot Circuits: Electronic Sensory Organs and Nerve Systems* which was published in 1960. The Plexiglas-covered dog called Cyber (pronounced See-ber) was a marvel for its day. **Figure 7** shows the dog without its plastic skin. A centrally-mounted battery supplied power.

Walking Movements

Looking at the four legs, each one has wheels, as they don't lift up for each step like real dogs. Notice the solenoids on the back legs; they act as brakes on the wheels to allow each leg to have traction to push backwards to propel the dog forward, or forward for reverse movement. The diagonal rods from the legs go to two reciprocating rocker arms that push the right pair of legs and the left pair of legs back and forth. Notice that both of the uncovered left legs are moved away from the dog's body, while the covered dog's left legs are shown brought into the body. The front legs really have no physical function other than making the dog look like it's walking.

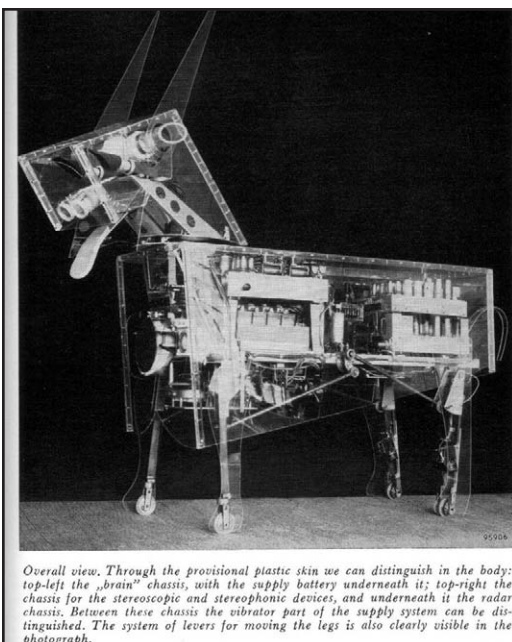


FIGURE 6. The Philips dog with Plexiglas shell.

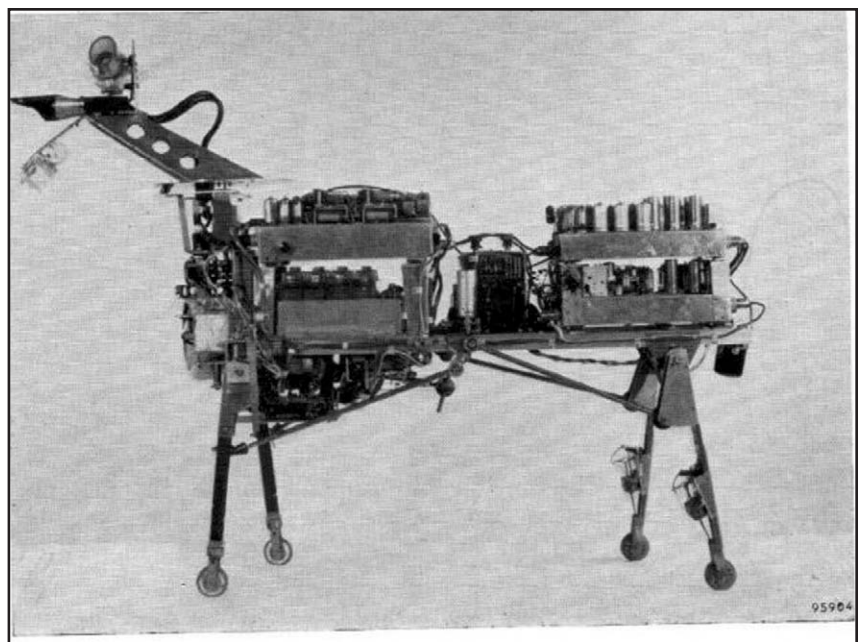


FIGURE 7. Philips dog with the shell removed.

You can easily see the four sets of electronic circuit chassis boxes that control all of the robot's functions. The eyes used the same photoelectric photocells that Walter used in his robots. Notice that the head and neck are mounted on a turntable so that the dog can see and track spots of light stereoscopically. The ears also act as sonic trackers with their dual microphones and are activated when the eyes detect no light. Cyber could actually respond to its name when called.

The Nose and Ears

The nose is actually a pair of temperature-sensitive Negative Temperature Coefficient (NTC) resistors. The circuit is tuned for the warmth of a hand or the heat of a hot dog. A hand held four inches in front of the nose will trigger the circuit after about one minute. A much hotter hot dog will trigger the circuit more quickly. The two resistors are positioned so a hot dog that is put in front of its nose to one side will cause the robot to turn its head in the direction of the hot dog. At that point, the motorized tongue is triggered as if to lick it.

The Acoustic Radar

The dark circle in the front chest of the robot is a small

loudspeaker used to project a sonar sound wave; the microphone receives the reflected sound to detect objects. The sound is that of a barking dog and lasts about three seconds. The circuitry is tuned to be sensitive to about one meter or 40 inches. If an object is detected in that space, the robot stops and then reverses. Side microswitches and the tail switch can contact surfaces to stop unwanted movement. If the tail switch or side switches contact something, the dog will reverse again.

Closing Thoughts

This robot dog was used as a behavioral experiment platform, yet the shape of a canine was used to exhibit certain animal stimuli/reaction responses. These early robotic creations were just stepping stones to more sophisticated machines to come later. Check out the two sites shown here and other sites for some very interesting robot history.

I am indebted to David Buckley who furnished a lot of the information and photographs on this unique robot, and many other robots that I've written about. David's site — **davidbuckley.net** — is always a treasure trove of interesting old robot information. The other is **cybernetic.zoo.com**, another great source of timelines of robot history. **SV**

New Electronics Merit Badge Lab
Easy to Learn, Easy to Teach

Includes: Workbook, Online Software, Online Video, Reusable Solderless Breadboard, electronic components, and solder Kit 9999.



Benefits of this hands-on Lab:

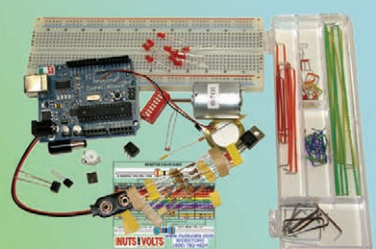
- Have a great time learning powerful new skills while earning a really fun Merit Badge.
- Everything is included, no difficult parts to find and buy.
- All the know-how and training material is provided in online videos, software, and a printed fully-illustrated workbook.
- A complete start to finish, hands-on learning experience.
- Earn a Tech Level 1 Certification as Electronics Technician with www.GSSTechEd.com

Be Worth More!
"LEARN ELECTRONICS"

tronix

1-800-422-1100
www.GSSTechEd.com/boyscouts.html

Puzzled by the Arduino?



An Arduino Workshop
Are you puzzled about the Arduino but finding it difficult to get all the pieces in one place?
Joe Pardue
SmileyMicros.com

Book and Kit Combo
\$124.95

Book \$44.95

Based on the Nuts&Volts Smileys Workshop, this set gives you all the pieces you need!

For more info on this and other great combos!
Please visit: <http://store.nutsvolts.com>

ROBO-LINKS

GPS MADE SIMPLE™



Linx GPS MODULE
HX1M-GPS-SG
LOT GA 0009

LinxTechnologies.com

LOCATE • TRACK • MAP • FIND • NAVIGATE

THE ORIGINAL SINCE 1994

PCB-POOL®

Beta LAYOUT

- Low Cost PCB prototypes
- Free laser SMT stencil with all Proto orders

WWW.PCB-POOL.COM

RobotShop.com



Pololu

Robotics & Electronics

WWW.POLOLU.COM



ALL ELECTRONICS CORPORATION

Electronic Parts & Supplies
Since 1967



AndyMark

Inspiring Mobility

www.andymark.com



ServoCenter

servo-center.com

- 16 servos, 16 I/O, 8 A/D
- USB, RS-232, TTL serial
- 14-bit servo control
- Built-in SC-BASIC Sequencer

ServoCenter 4.1 USB \$56.99

ServoCenter 4.1 USB MINI \$59.99



For the finest in robots,
parts, and services, go to
www.servomagazine.com
and click on **Robo-Links**.

Ultrasonic Ranging is EZ

- High acoustic power, auto calibration, & auto noise handling makes your job easy.
- Robust, compact, industrial (IP67) versions are available.

www.maxbotix.com



Das Blinkenboard

Not just for blinken LEDs.

Much Much More!

Complete kits available @
http://store.nutsvolts.com & http://store.servomagazine.com



INVEST in your BOT! #



HITEC

12115 Paine Street • Poway, CA 92064 • 858-748-6948 • **www.hitecrod.com**

superbrightleds.com

Component LEDs - LED Bulbs - LED Products

St. Louis, Missouri - USA Fast Online Ordering **superbrightleds.com**



To Advertise: Call 951-371-8497

ADVERTISER INDEX

All Electronics Corp.15, 81	Linx Technologies81	Sunstone CircuitsBack Cover
AndyMark33, 81	Lynxmotion, Inc.82	superbrightleds.com81
AP Circuits12	Maxbotix81	Technological Arts15
BaneBots63	PCB Pool39, 81	Vantec12
Cross The Road Electronics39	Pololu Robotics & Electronics3, 81	X-treme Geek7
Firgelli15	RobotShop, Inc17, 81	Yost/ServoCenter81, 83
GSS Tech Ed80	Robot Power33	
HiTec2, 81	Solarbotics/HVW63	

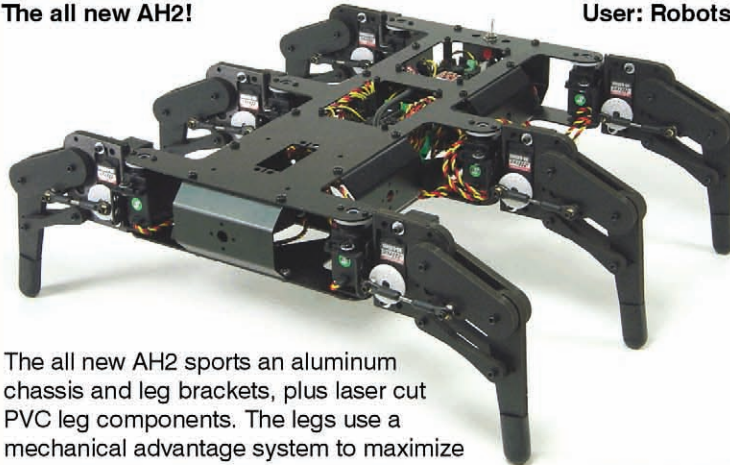


The Lynxmotion Servo Erector Set Imagine it... Build it... Control it!

Featured Robot

Coming Soon!
The all new AH2!

Youtube videos
User: Robots7



The all new AH2 sports an aluminum chassis and leg brackets, plus laser cut PVC leg components. The legs use a mechanical advantage system to maximize payload even when using inexpensive servos. This is 2DOF Hexapod, done right!

Black or clear
anodized finish!



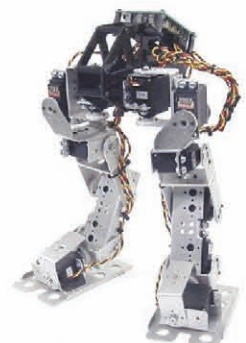
Biped Nick



Biped Pete



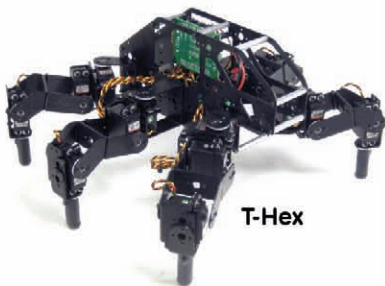
Biped Scout



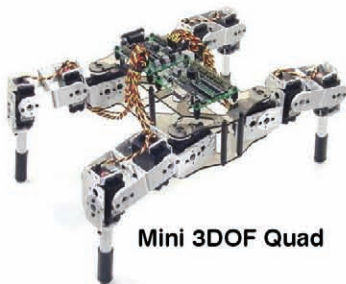
Biped 209



Walking Stick



T-Hex



Mini 3DOF Quad

With our popular Servo Erector Set you can easily build and control the robot of your dreams!

Our interchangeable aluminum brackets, hubs, and tubing make the ultimate in precision mechanical assemblies possible.



All New ARC-32 - \$99.95
32 Channel Servo/Microcontroller.
USB Programming port.
32 bit Hardware based math.
Program in BASIC, C, or ASM
Servo and Logic power inputs.
Sony PS2 game controller port.



Bot Board II - \$24.95
Carrier for Atom / Pro, BS2, etc.
Servo and Logic power inputs.
5vdc 250mA LDO Regulator.
Buffered Speaker.
Sony PS2 game controller port.
3 Push button / LED interface.



SSC-32 - \$39.95
32 Channel Servo Controller.
Speed, Timed, or Group moves.
Servo and Logic power inputs.
5vdc 250mA LDO Regulator.
TTL or RS-232 Serial Comms.
No better SSC value anywhere!

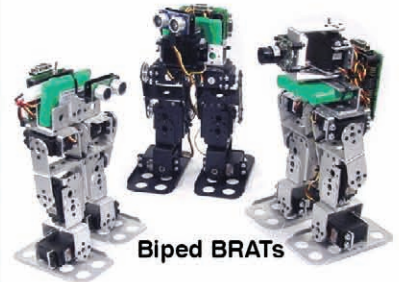
We also carry motors, wheels, hubs, batteries, chargers, servos, sensors, RC radios, pillow blocks, hardware, etc!



Visit our huge website to see our complete line of robots, electronics, and mechanical components.



CH3-R



Biped BRATs



Phoenix



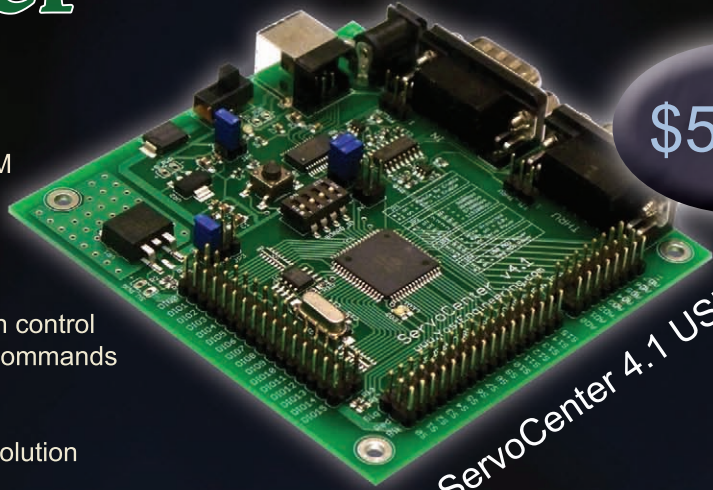
Images represent a fraction of what can be made! www.lynxmotion.com The SES now has over 200 unique components!

The Future of Servo Control is Calling...

ServoCenter™

Features

- USB, RS232, and TTL serial support
- 16 servos, 16 digital I/O, 8 analog inputs
- Built-in SC-BASIC Sequencer with EEPROM
- Sequencer allows stand-alone operation
- 64 scene presets stored in EEPROM
- Presets instantly loaded or cross-faded
- Built-in configurable smoothing algorithm
- Independent, simultaneous speed & position control
- Scaled, percentage, and group movement commands
- Timed movement commands
- Max, min, & startup position settings
- Ultra-precise 0.05425µS jitter-free pulse resolution

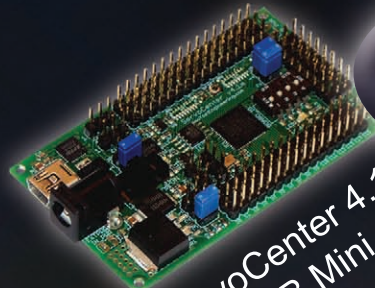


\$56.99

ServoCenter 4.1 USB

Flexibility

- User upgradeable firmware
- Upload your own firmware with bootloader
- Watchdog timer for failsafe operation
- Over-current, over-temperature, polarity protection
- Internal regulator, external power, or battery power options
- Supports 4.8/6.0V regulated servo supply voltages at 5 Amps
- Each digital I/O and analog input has supply pins
- Direct serial, activeX control, or Win32 DLL communication
- Programming examples in 10+ languages
- Windows, Linux, Mac OSX compatible

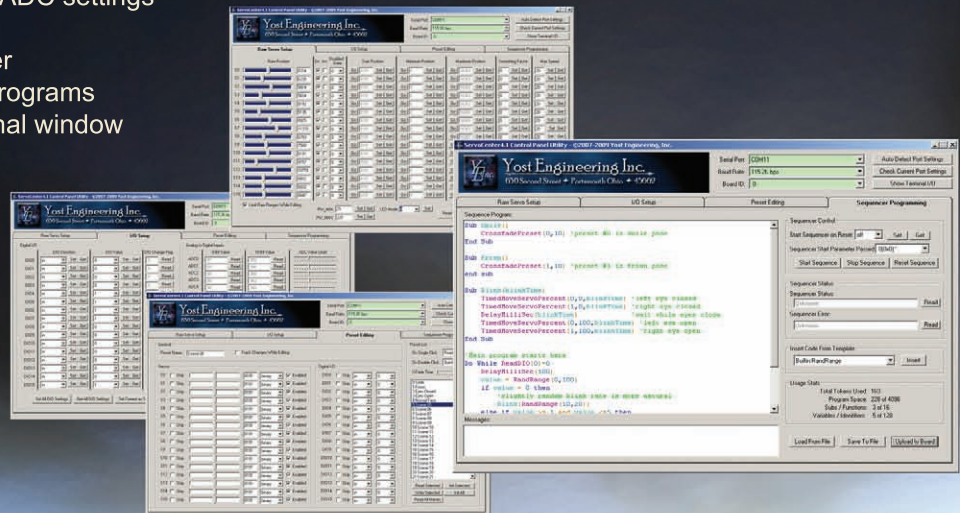


\$59.99

ServoCenter 4.1
USB Mini

Free Control Panel Software

- Easy editing and configuration of servo settings
- Configure and set up digital I/O & ADC settings
- Edit scene presets
- Program the SC-BASIC sequencer
- Upload and run your SC-BASIC programs
- Debug & communicate with terminal window



www.Servo-Center.com

Yost Engineering Inc.



NO MATTER WHAT THE IDEA YOUR PCB PROTOTYPES SHOULD BE THE EASY PART

QUOTE & ORDER PCBs ONLINE AT WWW.SUNSTONE.COM OR CALL 1-800-228-8198



THE EASIEST PCB COMPANY TO DO BUSINESS WITH



ValueProto™



PCBexpress®



Full Feature

Sunstone Circuits® pioneered the online ordering of printed circuit boards and is the leading PCB solutions provider with more than 35 years of experience in delivering quality prototypes and engineering software. With this knowledge and experience, Sunstone is dedicated to improving the PCB prototyping process from quote to delivery (Q2D®).

Did You Know? Sunstone Offers:

- Controlled impedance testing
- Free 25-point design review
- Online Quote & Order
- Over 99% on-time or early delivery
- Fine lines and spacing [.003]
- Free shipping & no NRE's
- PCB123® design software
- Best PCBs in the industry
- RoHS compliant finishes
- Flex / Rigid Flex Boards
- RF / Exotic Materials
- Live customer support 24/7/365